

## CHƯƠNG I

## MAVEN

### 1. Khái niệm :

Maven là một phần mềm công cụ, xuất bản bởi Apache Software Foundation, được sử dụng để xây dựng và quản lý các dự án một cách tự động, dựa trên một khái niệm là POM (viết tắt của Project Object Model).

POM là một đơn vị nền tảng của Maven, đó là một file XML chứa đựng các thông tin và cấu hình của dự án. Những thông tin và cấu hình này sẽ được Maven sử dụng để xây dựng nên cấu trúc của dự án.

Maven được sử dụng chính trong Java, nhưng bên cạnh đó còn hỗ trợ cho các ngôn ngữ khác như PHP, C#, Ruby, Scala...

### 2. Chức năng :

Mục đích chính của Maven là giúp cho các nhà phát triển phần mềm có thể triển khai dự án hoàn thành trong thời gian ngắn nhất và có hiệu quả. Maven có thể quản lý việc xây dựng dự án, báo cáo và tài liệu hóa thông tin dự án.

Ngoài chức năng xây dựng và quản lý thông tin các dự án, Maven còn cho phép tự động download các thư viện và các plug-ins từ một hay nhiều nơi từ trên mạng về. Các nơi này được gọi là Maven remote repositories (các kho chứa). Mặc định thì Maven sẽ lấy thư viện từ remote repository của Maven tại địa chỉ <http://repo1.maven.org/maven2>.

Các thư viện này sau khi được lấy về từ trên mạng sẽ được lưu trữ tại máy tính để sử dụng cho những lần sau. Về nguyên tắc, khi chạy, Maven sẽ tìm các thư viện ở kho chứa local trong máy tính trước, nếu không có thì sẽ lấy từ trên mạng. Vì vậy, chỉ có lần đầu tiên build chương trình hoặc khi dự án cần sử dụng thư viện mới thì máy tính phải nối kết nối với mạng Internet, còn những lần tiếp sau đó thì có thể chạy offline.

### 3. Cài đặt :

Hiện tại Maven cung cấp 2 phiên bản Maven 2 và Maven 3. Trong bài báo cáo này sử dụng phiên bản Maven2. Ta có thể download Maven về tại địa chỉ <http://maven.apache.org/>.

Sau khi download Maven về máy tính và giải nén, việc tiếp theo chúng ta nên cấu hình local repository cho Maven.

Đầu tiên, ta tạo một thư mục con có tên là m2repository trong thư mục gốc của maven, sau đó mở file setting.xml trong thư mục conf của Maven và tìm đến vị trí thẻ <localRepository>, thiết lập giá trị như sau:

```
<localRepository>đường dẫn đến thư mục m2repository</localRepository>
```

Để chạy Maven, yêu cầu máy tính phải cài đặt sẵn JDK 5 hoặc cao hơn. Để thuận tiện cho việc chạy Maven, ta tạo ra các biến môi trường với giá trị như sau :

JAVA\_HOME= Đường dẫn đến thư mục gốc của JDK

M2\_HOME= Đường dẫn đến thư mục gốc của Maven 2

CATALINA\_HOME= Đường dẫn đến thư mục gốc của Tomcat

M2\_REPOSITORY= %M2\_HOME%\m2repository

Sau đó thêm vào PATH các giá trị sau:

%JAVA\_HOME%\bin;%M2\_HOME%\bin;%PATH%;C:\WINDOWS\system32

Sau khi thiết lập biến môi trường xong, mở cửa sổ cmd và gõ lệnh mvn -version để kiểm thử kết quả cài đặt.

Đến đây đã hoàn thành việc cài đặt Maven.

### 4. Cách sử dụng Maven:

Khi làm việc với Maven, ta có thể tạo ra nhiều loại dự án khác nhau như các ứng dụng Web, các ứng dụng cho Windows, hay cả những ứng dụng cho điện thoại di động sử dụng hệ điều hành Android.

Ngoài ra, Maven còn có thể làm việc trên nhiều IDE như Eclipse, NetBeans, IntelliJIDEA, JBuilder... Trong bài báo cáo này sử dụng Eclipse IDE để minh họa.

Nhưng trước khi đi vào tìm hiểu cách làm thế nào để sử dụng Maven xây dựng nên một dự án, chúng ta sẽ tìm hiểu qua về hai khái niệm Maven Dependencies và Maven Plug-ins.

#### a) Maven Dependencies:

Maven cung cấp một cơ chế được gọi là cơ chế phụ thuộc. Cơ chế này bao gồm 2 kiểu. Thứ nhất là sự phụ thuộc giữa dự án với các thư viện, framework. Thứ hai là sự phụ thuộc giữa các dự án với nhau. Thông thường, các dependency này là các đoạn mã lệnh đã được đóng gói thành các file thực thi. Như vậy, nhờ cơ chế này, Maven đã giải quyết được vấn đề tái sử dụng và kế thừa mã lệnh. Từ đó tiết kiệm được rất nhiều thời gian xây dựng chương trình.

Cơ chế phụ thuộc có tính bắc cầu. Giả sử dự án A phụ thuộc dự án B, dự án B lại phụ thuộc dự án C thì khi đó dự án C cũng sẽ phụ thuộc vào dự án A. Nhưng cũng có những trường hợp thì tính bắc cầu không thực sự cần thiết thì sao?... Maven đã cung cấp một ràng buộc được gọi là phạm vi phụ thuộc (Dependency Scope).

Dependency Scope bao gồm 4 phạm vi dưới đây:

+compile : Là phạm vi mặc định, được sử dụng khi không có bất kì phạm vi được chỉ định. Khi sử dụng phạm vi này, sự phụ thuộc có phạm vi phụ thuộc này sẽ được lan truyền qua tất cả các dự án phụ thuộc. Thường được dùng cho các thư viện cần thiết sử dụng cho việc biên dịch

+provided : Cũng gần giống như compile nhưng không có tính bắc cầu

+runtime : Thường được dùng đối với các thư viện chỉ cần thiết cho quá trình thực thi, không được yêu cầu trong quá trình biên dịch, có tính chất bắc cầu

+test : Thường được dùng cho các thư viện phục vụ cho việc testing, không có tính chất bắc cầu

Các phụ thuộc này được khai báo trong pom.xml và sau đây là qui cách khai báo chuẩn:

```
<dependency>
    <groupId>groupId</groupId>
    <artifactId>artifactId</artifactId>
    <version>version</version>
    <scope>scope</scope>
</dependency>
```

Ví dụ cụ thể về sự phụ thuộc :

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
</dependency>
```

Ở trên, ta đã sử dụng thư viện junit cho dự án, đây là thư viện được dùng cho việc testing nên scope được khai báo là test.

Tiếp sau đây là 1 file pom.xml đơn giản :

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>groupId</groupId>
    <artifactId>artifactId</artifactId>
    <version>version</version>
    <packaging>packaging</packaging>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
```

</dependencies>

</project>

Trong đó :

- project : Thành phần cấp cao nhất của file pom.xml
- modelVersion : Phiên bản mà object model sử dụng. Mặc định Maven 2 chỉ định luôn là 4.0.0
- groupId : Tên tổ chức hoặc nhóm tạo ra dự án. Ngoài ra đây cũng là tên package mặc định của dự án. Dựa vào groupId, Maven sẽ tạo ra một khu vực dành riêng cho tổ chức hoặc nhóm trong local repository ở máy tính cục bộ. Tại đây sẽ lưu lại toàn bộ thông tin, kết quả thực thi của dự án bao gồm các file cấu hình XML, JAR, WAR, EAR...groupId nên tuân theo qui ước đặt tên package trong Java
- packaging: Chỉ định loại file thực thi của dự án sau khi được biên dịch ( jar, war, ear...)
- artifactId : Tên của dự án.
- version : Phiên bản của dự án. Giá trị mặc định là 1.0-SNAPSHOT
- dependencies: Bao gồm tất cả các thư viện mà dự án sử dụng
- dependency : Chi tiết một thư viện mà dự án sử dụng

#### **b) Maven Plug-ins:**

Nếu Maven Dependencies là các thư viện, framework cung cấp chức năng cho dự án thì Maven Plug-ins lại là công cụ để thực thi các thư viện, framework đó. Tất cả công việc đều được hoàn thành bởi các plug-in này.

Maven Plug-ins được chia làm 2 nhóm : build plug-ins và reporting plug-ins. Hiện nay có rất nhiều plug-in mà Maven cung cấp, chi tiết về các plug-in này chúng ta có thể tham khảo tại website chính thức của Maven : <http://maven.apache.org/plugins/index.html>

Tiếp sau đây, chúng ta sẽ tìm hiểu cách sử dụng các plug-in phổ biến hay được sử dụng nhất khi làm việc với Maven.

#### **c) Tạo dự án với Maven:**

Để tạo một dự án với Maven, chúng ta sẽ sử dụng plug-in có tên archetype. Đây là tập hợp các template các loại dự án khác nhau được gọi là các archetype. Plug-in này cung cấp 4 chức năng sau đây :

- +archetype:create : Tạo một dự án từ một template có sẵn
- +archetype:generate : Tạo một dự án từ một template có sẵn nhưng sẽ cho người dùng lựa chọn loại template mong muốn.
- + archetype:create-from-project : Tạo ra một template từ một dự án có sẵn
- + archetype:crawl: Tìm kiếm repository cho các template và cập nhật cho chúng

- Tạo dự án sử dụng archetype:create :

-Để tạo dự án cho Windows :

mvn archetype:create -DgroupId=groupId -DartifactId=artifactId -Dversion=version

-Để tạo dự án cho Web :

mvn archetype:create -DgroupId=groupId -DartifactId=artifactId -Dversion=version  
DarchetypeArtifactId=maven-archetype-webapp

Chú ý rằng thêm vào câu lệnh DarchetypeArtifactId=maven-archetype-webapp để chỉ cho Maven biết ta muốn tạo một dự án Web.

- Tạo dự án sử dụng archetype:generate :

Cú pháp như sau : mvn archetype:generate

Kết quả sau lệnh này là một danh sách rất nhiều template dự án được đánh số thứ tự, việc còn lại của chúng ta là chọn template mà mình mong muốn cho dự án.

Tùy theo loại template mà ta chọn thì khuôn mẫu thu được sẽ khác nhau nhưng hầu hết đều có điểm chung như sau :

Thư mục	Ý nghĩa
Thư mục gốc	Chứa file pom.xml và toàn bộ thư mục con
src/main/java	Nơi chứa các lớp Java cho dự án
src/main/resources	Nơi chứa các file tài nguyên cần thiết cho dự án như các file thuộc tính, file cấu hình
src/test/java	Nơi chứa các lớp cho việc testing
src/test/resources	Nơi chứa các tài nguyên cần thiết cho testing

Hình 1.1 - Cấu trúc chung của một dự án Maven

Maven hỗ trợ làm việc trên rất nhiều framework như Hibernate, Spring MVC, Struts,... Khi dự án cần phát triển trên các framework nào, chúng ta chỉ cần thêm vào file pom.xml các dependencies các framework đó. Việc thêm các dependencies chúng ta nên tham khảo trên trang chủ của Maven hoặc trên các trang tutorial trên Web như <http://www.mkyong.com>, <http://www.roseindia.net> ...

Sau đây là danh sách các dependencies sử dụng cho Hibernate và Spring :

```
<!-- Hibernate framework dependencies -->
<dependency>
    <groupId>hibernate</groupId>
    <artifactId>hibernate3</artifactId>
    <version>3.2.3.GA</version>
</dependency>
<dependency>
    <groupId>hibernate-annotations</groupId>
    <artifactId>hibernate-annotations</artifactId>
    <version>3.3.0.GA</version>
</dependency>
<dependency>
    <groupId>hibernate-commons-annotations</groupId>
    <artifactId>hibernate-commons-annotations</artifactId>
    <version>3.0.0.GA</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.6.1</version>
</dependency>
<dependency>
    <groupId>cglib</groupId>
    <artifactId>cglib</artifactId>
    <version>2.2</version>
</dependency>
<dependency>
```

```
        <groupId>asm</groupId>
        <artifactId>asm</artifactId>
        <version>3.0</version>
    </dependency>
    <dependency>
        <groupId>dom4j</groupId>
        <artifactId>dom4j</artifactId>
        <version>1.6.1</version>
    </dependency>
    <dependency>
        <groupId>commons-logging</groupId>
        <artifactId>commons-logging</artifactId>
        <version>1.1.1</version>
    </dependency>
    <dependency>
        <groupId>commons-collections</groupId>
        <artifactId>commons-collections</artifactId>
        <version>3.2.1</version>
    </dependency>
    <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-io</artifactId>
        <version>1.3.2</version>
    </dependency>
    <dependency>
        <groupId>commons-lang</groupId>
        <artifactId>commons-lang</artifactId>
        <version>2.3</version>
    </dependency>
    <dependency>
        <groupId>antlr</groupId>
        <artifactId>antlr</artifactId>
        <version>2.7.7</version>
    </dependency>
<!-- Java Persistence API dependencies -->
    <dependency>
        <groupId>javax.persistence</groupId>
        <artifactId>persistence-api</artifactId>
        <version>1.0</version>
    </dependency>
    <dependency>
        <groupId>javax.transaction</groupId>
        <artifactId>jta</artifactId>
        <version>1.1</version>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>servlet-api</artifactId>
        <version>2.5</version>
    </dependency>
```

```

        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.1.2</version>
            <scope>runtime</scope>
        </dependency>
<!-- MySQL database driver dependency -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.9</version>
        </dependency>
<!-- Spring framework dependencies -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>org.springframework.core</artifactId>
            <version>3.0.1.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>org.springframework.web</artifactId>
            <version>3.0.1.RELEASE </version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>org.springframework.web.servlet</artifactId>
            <version>3.0.1.RELEASE </version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring</artifactId>
            <version>2.5.6</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-orm</artifactId>
            <version>2.5.1</version>
        </dependency>

```

Sau khi đã thêm đầy đủ các thư viện cần thiết cho dự án vào pom.xml. Việc còn lại là chạy Maven. Từ cửa sổ cmd, ta di chuyển đến thư mục gốc của dự án (nơi chứa file pom.xml) , sử dụng lệnh :

```
mvn install
```

Lệnh này sẽ download các thư viện cần thiết về local repository để sử dụng cho những lần sau hoặc cho những dự án khác, đồng thời cài đặt chúng vào classpath của dự án, chạy các file test (nếu có), biên dịch source code thành các file thực thi đồng thời cũng đưa các file này vào local repository. Đây cũng là lệnh hay được sử dụng nhất khi làm việc với Maven.

Cuối cùng, để có thể import dự án vào Eclipse IDE, ta sử dụng lệnh :

```
mvn eclipse:eclipse
```

Dưới đây là danh sách một số lệnh cơ bản khác và các chức năng chính của chúng :

- +mvn validate : Kiểm tra lại thông tin về dự án
- +mvn compile : Biên dịch mã nguồn của dự án thành các file .class
- +mvn test : Chạy các file test chương trình của dự án
- +mvn package : Đóng gói chương trình thành các file thực thi (.jar, .war)
- +mvn verify : Kiểm tra sự hợp lệ của tất cả các gói trong chương trình
- +mvn deploy : Được sử dụng trong các dự án web để đưa file war của chương trình đến web server như Glashfish, Tomcat, WebLogic, JRun...
- +mvn clean : Xóa toàn bộ kết quả đã được biên dịch bằng lệnh mvn compile
- +mvn site : Tạo document site cho dự án

### 5. Deploy dự án Maven Web đến Tomcat:

Trước hết chúng ta phải cấu hình web server trong file setting.xml nằm trong thư mục conf của Maven:

```
<server>
  <id>tomcat</id>
  <username>username</username>
  <password>*****</password>
</server>
```

Trong đó :

- id: Tên server Tomcat
- username : Tên đăng nhập vào Tomcat
- password : Mật khẩu đăng nhập vào Tomcat

Sau khi đã cấu hình xong server, trong file pom.xml của dự án, ta thêm vào như sau :

```
<build>
  <plugins>
    <plugin>
      <groupId>org.codehaus.cargo</groupId>
      <artifactId>cargo-maven2-plugin</artifactId>
      <version>1.0.3</version>
      <configuration>
        <type>standalone</type>
        <home>${CATALINA_HOME}</home>
        <properties>
          <cargo.jvmargs>
            -Xdebug-Xrunjdwp:server=y,transport=dt_socket,address=8000,suspend=n
          </cargo.jvmargs>
          <cargo.tomcat.manager.url>
            http://localhost:8080/manager
          </cargo.tomcat.manager.url>
          <cargo.hostname>localhost</cargo.hostname>
          <cargo.servlet.port>8080</cargo.servlet.port>
          <cargo.remote.username>username</cargo.remote.username>
          <cargo.remote.password>***** </cargo.remote.password>
        </properties>
        <deployer>
          <type>installed</type>
          <deployables>
            <deployable>
              <groupId>groupId</groupId>
              <artifactId>artifactId</artifactId>
```

```

        <type>type</type>
      </deployable>
    </deployables>
  </deployer>
  <container>
    <containerId>tomcat6x</containerId>
    <type>installed</type>
    <home>${CATALINA_HOME}</home>
  </container>
</configuration>
</plugin>
</plugins>
</build>

```

Ở trên, chúng ta sử dụng plug-in cargo để làm nhiệm vụ deploy các file war đến web server.

Trong thẻ deployable :

+groupId : groupId của dự án

+artifactId : artifactId của dự án

+type : Thông thường có giá trị là war hoặc ear tùy thuộc vào dự án

Sau khi đã cấu hình xong, việc đầu tiên là chúng ta phải chạy server Tomcat bằng lệnh sau đây : %CATALINA\_HOME%\bin\catalina.bat jpda start

Để deploy chương trình đến Tomcat ta sử dụng lệnh sau : mvn cargo:deploy

Như vậy, chúng ta đã tìm hiểu xong những nét cơ bản về Maven. Có thể nói, khi làm việc với Maven, công việc build chương trình trở nên đơn giản và dễ dàng hơn rất nhiều. Cũng nói thêm, trong lĩnh vực phát triển phần mềm, bên cạnh Maven còn có một công cụ build dự án từ lâu cũng đã rất nổi tiếng , đó là Ant. Nhưng kịch bản của Ant lại phức tạp hơn Maven, hơn nữa Ant không có khả năng sử dụng lại mã nguồn và tốc độ thực thi chậm hơn Maven. Chính vì thế mà Maven ngày càng được sử dụng phổ biến trong các dự án Java.



## CHƯƠNG II

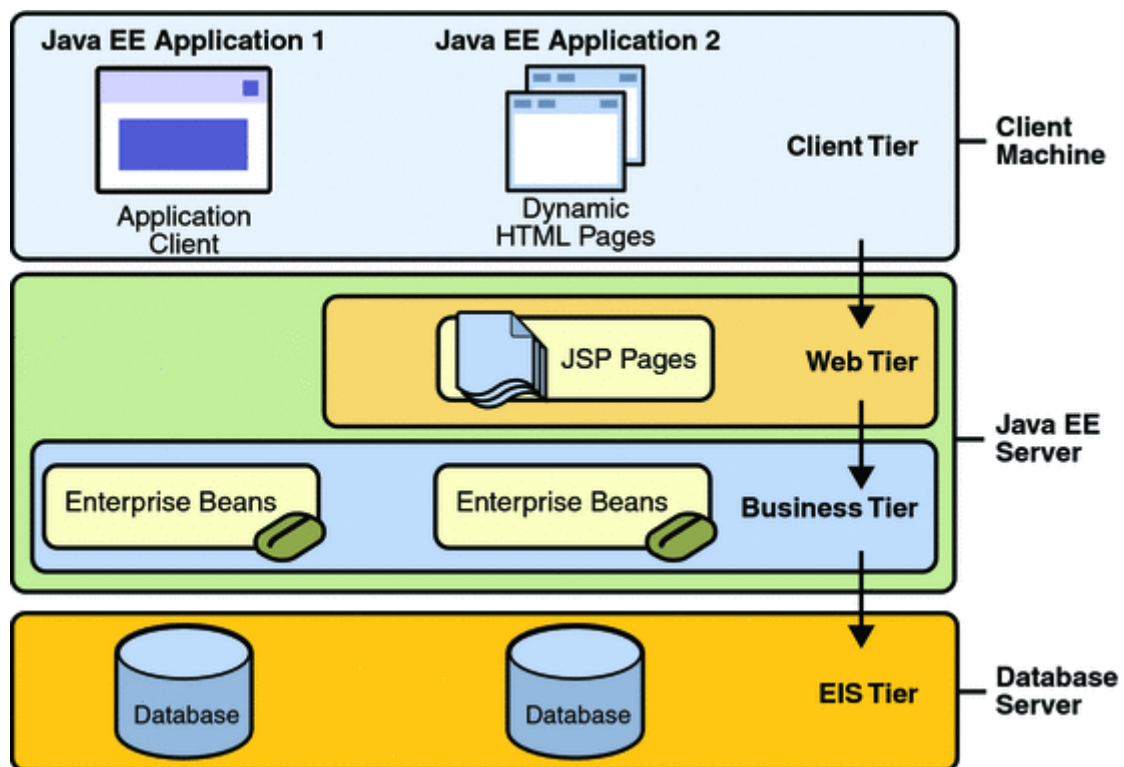
## HIBERNATE FRAMEWORK

**I. Tổng quan về Hibernate :****1. Định nghĩa :**

Hibernate là framework ánh xạ đối tượng-quan hệ rất phổ biến cho ngôn ngữ lập trình Java và là thành phần quan trọng của J2EE EJB (Enterprise Java Bean), được phát triển bởi Red Hat. Hibernate cho phép ánh xạ các lớp Java đến các bảng trong cơ sở dữ liệu, cung cấp truy vấn trên cơ sở dữ liệu một cách linh hoạt, chặt chẽ. Từ đó làm giảm đáng kể thời gian phát triển nhưng vẫn đảm bảo độ tin cậy của các sản phẩm phần mềm.

Người ta gọi các lớp Java này là các đối tượng persistence. Một đối tượng được gọi là persistence khi nó được tạo ra và lưu lại trong một quá trình xử lý nào đó, khi quá trình đó kết thúc hay khi hệ thống thông tin shutdown thì các đối tượng này không bị mất đi mà nó được lưu lại trên thiết bị lưu trữ dưới dạng khác.

Hibernate không phải là giải pháp tốt nhất cho các ứng dụng dữ liệu trung tâm, nơi mà các chức năng của ứng dụng được thực hiện chủ yếu bằng cách sử dụng các stored-procedure được tạo ra ngay trong cơ sở dữ liệu mà Hibernate thực sự trở nên rất mạnh trong các ứng dụng hướng đối tượng với mô hình đa tầng (2-tier hay n-tier).

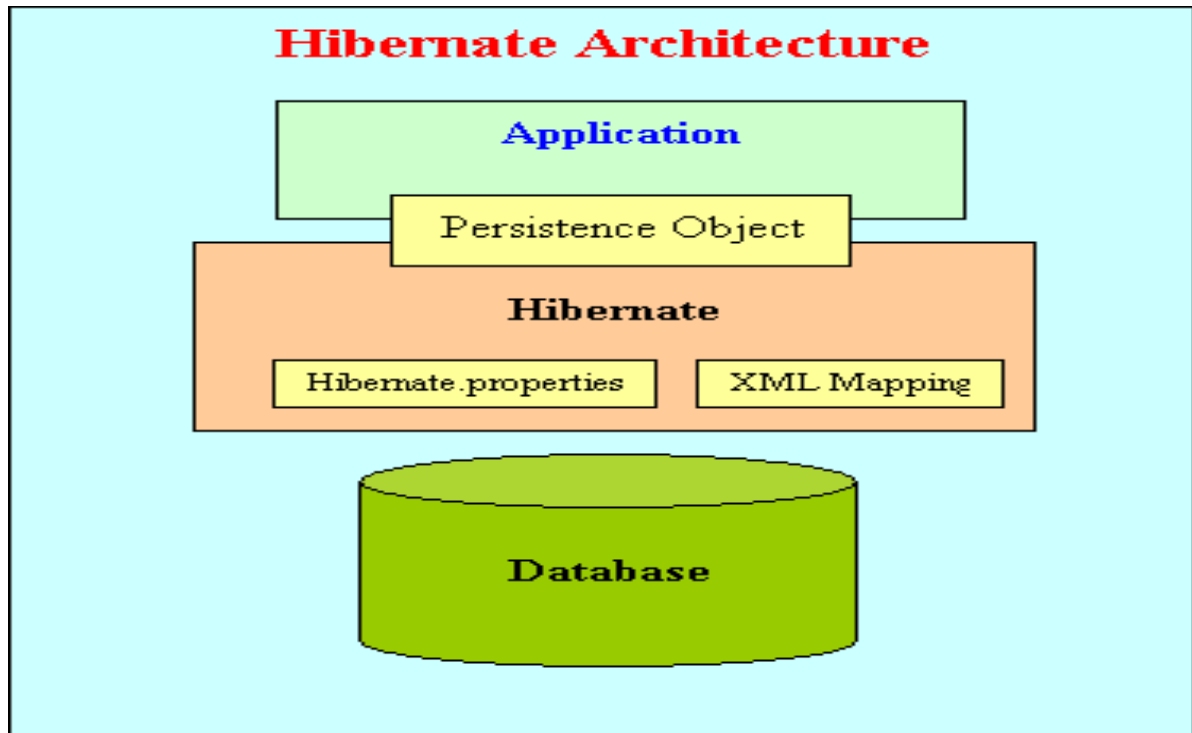


Hình 2.1 - Mô hình 3-tier

Hibernate cho phép trong suốt persistence (khả năng lưu dữ liệu trực tiếp xuống hệ quản trị cơ sở dữ liệu thông qua các đối tượng trong ngôn ngữ lập trình), từ đó ứng dụng có thể chuyển đổi qua bất kỳ hệ quản trị cơ sở dữ liệu nào. Có thể nói đây là một trong những điểm mạnh của Hibernate.

Hibernate có thể sử dụng trong các ứng dụng cho Swing, ứng dụng Servlet, JSP..hay ứng dụng J2EE sử dụng các EJB session bean.

## 2.Kiến trúc của Hibernate:



Hình 2.2 – Kiến trúc Hibernate

Để sử dụng được Hibernate, chúng ta phải tạo ra các lớp Java, mỗi lớp là thể hiện của một bảng được lưu trong cơ sở dữ liệu. Hibernate sẽ ánh xạ các thuộc tính của lớp với các cột của bảng dữ liệu. Nhờ đó, Hibernate có thể thực hiện các phép toán trên cơ sở dữ liệu như select, insert, update và delete các bộ trong bảng thông qua các câu truy vấn.

Nhìn chung Hibernate bao gồm 3 thành phần chính :

- Quản lý kết nối : Cung cấp quản lý kết nối cơ sở dữ liệu một cách hiệu quả bởi lẽ kết nối cơ sở dữ liệu là một công đoạn rất tốn kém vì nó yêu cầu chiếm nhiều tài nguyên để mở và đóng kết nối.
- Quản lý giao dịch : Cung cấp cho người sử dụng có thể thực hiện một hay nhiều câu lệnh truy vấn tại một thời điểm.
- Ánh xạ đối tượng - quan hệ : Là kỹ thuật cho phép ánh xạ từ một mô hình đối tượng đến một mô hình dữ liệu hay nói cách khác là ánh xạ từ lớp Java đến bảng dữ liệu.

### 3.Tính năng :

- Cung cấp Hibernate Query Language : Là ngôn ngữ truy vấn dữ liệu khá mạnh . Nó hầu như rất giống với SQL và có phân biệt chữ hoa chữ thường loại trừ tên của các lớp Java và thuộc tính của nó.
- Cung cấp Hibernate Criteria Query API : Sử dụng để tạo ra các câu truy vấn động. Đây là một trong những phương pháp viết câu truy vấn mà không cần sử dụng đến HQL. Cung cấp đầy đủ các phép chiếu (projection), phép kết (aggregation), truy vấn lồng nhau (subselects).
- Hỗ trợ cho Eclipse bao gồm nhiều plug-in như trình soạn thảo XML cho việc ánh xạ, truy vấn cơ sở dữ liệu, đặc biệt cung cấp 2 kỹ thuật có tác dụng gần như trái ngược nhau. Kỹ thuật thứ nhất có tên gọi “schema reverse engineering”, kỹ thuật này cho phép sinh mã lệnh các đối tượng Java, các lớp truy suất cơ sở dữ liệu(DAO – Date Access Object), các file cấu hình XML từ giản đồ của cơ sở dữ liệu đã có trong hệ quả trị cơ sở dữ liệu. Kỹ thuật thứ hai có tên “schema forward engineering”, kỹ thuật này lại cho phép phát sinh cơ sở dữ liệu từ các đối tượng Java, các file cấu hình XML. Với hai kỹ thuật này, người lập trình sẽ giảm bớt được thời gian viết mã lệnh cho việc cấu hình cơ sở dữ liệu một cách thủ công.
- Giảm thời gian phát triển sản phẩm phần mềm : Khả năng tái sử dụng mã lệnh do mang đậm tính chất hướng đối tượng của ngôn ngữ lập trình Java như cung cấp tính đóng gói , kế thừa , đa hình...

● Cung cấp các toán hạng làm việc với đối tượng persistence : create() , merge() , save() , saveOrUpdate() , saveOrUpdateCopy() ...

●Hibernate XML mapping : Là kỹ thuật ánh xạ đối tượng – quan hệ, trong đó thông tin về các bảng trong cơ sở dữ liệu có thể được biểu diễn trong các file XML. Đây là kỹ thuật được sử dụng phổ biến nhất hiện nay. Ưu điểm của phương pháp này là làm cho hệ thống dễ bảo trì, khả năng dễ dàng thay đổi khi có sự thay đổi trong cơ sở dữ liệu.

●Hibernate annotations : Cũng là một kỹ thuật ánh xạ đối tượng – quan hệ, trong đó thông tin về các bảng trong cơ sở dữ liệu được biểu diễn ngay trong các lớp Java bằng cách sử dụng các annotation. Phương pháp này có ưu điểm là dễ viết, nhưng lại làm cho hệ thống thêm rối , từ đó hệ thống sẽ khó bảo trì hơn. Do vậy, trên thực tế ít được sử dụng. Chúng ta sẽ tìm hiểu sâu hơn về 2 kỹ thuật ánh xạ này trong những phần sau của bài báo cáo.

## II.Cấu hình Hibernate :

### 1.Tạo kết nối cơ sở dữ liệu :

Để Hibernate có thể kết nối đến cơ sở dữ liệu, chúng ta phải cấu hình kết nối cho Hibernate. Thông thường, người ta thực hiện trong một file XML có tên là hibernate.cfg.xml. Mặc định, khi hệ thống khởi động, Hibernate sẽ tìm và đọc các file này để thực hiện kết nối đến hệ quản trị cơ sở dữ liệu.

Ví dụ về một file hibernate.cfg.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0/EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">${jdbc.driverClassName}</property>
    <property name="hibernate.connection.url">${jdbc.url}</property>
    <property name="hibernate.connection.username">${jdbc.username}</property>
    <property name="hibernate.connection.password">${jdbc.password}</property>
    <property name="hibernate.dialect">${jdbc.databaseDialect}</property>
    <property name="hibernate.hbm2ddl.auto">update/create/ create-drop </property>
    <property name="hibernate.show_sql">true/false</property>
    <mapping class="vn.edu.ptithcm.pim.dom.Project"/>
    <mapping resource="pim.hbm.xml" />
  </session-factory>
</hibernate-configuration>
```

Trong đó :

-\${jdbc.driverClassName} : Driver của hệ quản trị cơ sở dữ liệu mà chương trình sử dụng. Hiện tại, Hibernate hỗ trợ làm việc trên hầu hết các hệ quản trị cơ sở dữ liệu như MS SQL Server, Oracle, MySQL Server, DB2, Postgresql...

□ My SQL - com.mysql.jdbc.Driver

□ SQLServer - com.microsoft.jdbc.sqlserver.SQLServerDriver

□ Oracle - oracle.jdbc.driver.OracleDriver

□ DB2 - com.ibm.db2.jdbc.app.DB2Driver

□ Postgresql - org.postgresql.Driver

-\${jdbc.url} : Địa chỉ url chỉ đến cơ sở dữ liệu.

□ My SQL - jdbc:mysql://localhost:3306/ databaseName

□ SQLServer - jdbc:sqlserver://localhost:1433;databaseName=databaseName

;integratedSecurity=true;

□ Oracle - jdbc:oracle:thin:@localhost:1525: databaseName

□ DB2 - jdbc:db2:databaseName

□ Postgresql - jdbc:postgresql://localhost:5432/ databaseName

-\${jdbc.username} : Tên đăng nhập vào hệ quản trị cơ sở dữ liệu.

-\${jdbcpassword} : Mật khẩu đăng nhập vào hệ quản trị cơ sở dữ liệu.

-\${jdbcdialect} : Database Dialect của hệ quản trị cơ sở dữ liệu. Chỉ cho Hibernate biết chương trình muốn làm việc trên hệ quản trị cơ sở dữ liệu nào. Dưới đây là danh sách các dialect phổ biến :

- DB2 - org.hibernate.dialect.DB2Dialect
- HypersonicSQL - org.hibernate.dialect.HSQLDialect
- Informix - org.hibernate.dialect.InformixDialect
- Ingres - org.hibernate.dialect.IngresDialect
- Interbase - org.hibernate.dialect.InterbaseDialect
- Pointbase - org.hibernate.dialect.PointbaseDialect
- PostgreSQL - org.hibernate.dialect.PostgreSQLDialect
- Mckoi SQL - org.hibernate.dialect.MckoiDialect
- Microsoft SQL Server - org.hibernate.dialect.SQLServerDialect
- MySQL - org.hibernate.dialect.MySQLDialect
- Oracle - org.hibernate.dialect.OracleDialect
- Oracle 9 - org.hibernate.dialect.Oracle9Dialect
- Progress - org.hibernate.dialect.ProgressDialect
- FrontBase - org.hibernate.dialect.FrontbaseDialect
- SAP DB - org.hibernate.dialect.SAPDBDialect
- Sybase - org.hibernate.dialect.SybaseDialect
- Sybase Anywhere - org.hibernate.dialect.SybaseAnywhereDialect

-*hibernate.hbm2ddl.auto* : Cho phép tự động tạo(create), cập nhật(update) cơ sở dữ liệu khi hệ thống khởi động hay tạo cơ sở dữ liệu khi khởi động rồi xóa khi hệ thống shutdown(create-drop) . Chú ý nên thận trọng khi sử dụng create-drop. Đây là giá trị không được khuyến khích sử dụng.

-*hibernate.show\_sql* : Cho phép hiển thị tất cả câu lệnh truy vấn trên cơ sở dữ liệu trong quá trình làm việc với ứng dụng hay không. Tính năng này thường được sử dụng khi dự án đang trong giai đoạn phát triển, nhờ đó ta có thể kiểm tra được chương trình chạy như thế nào. Giá trị mặc định của thuộc tính này là false.

-*Thẻ mapping* : Chỉ định kỹ thuật ánh xạ đối tượng – quan hệ được sử dụng. Ví dụ trên sử dụng cả hai phương pháp ánh xạ đã được đề cập ở phần trên. Trong đó “class” - sử dụng kỹ thuật Annotations và “resource” - sử dụng kỹ thuật dựa trên cấu hình XML. Lớp Project sẽ là thể hiện của bảng Project trong cơ sở dữ liệu. Còn file pim.hbm.xml sẽ chứa toàn bộ mô hình của cơ sở dữ liệu được biểu diễn dưới dạng XML.

Ngoài những thuộc tính nêu trên, Hibernate còn cung cấp rất nhiều thuộc tính khác (ước tính có khoảng gần 60 thuộc tính khác nhau), trong đó có nhiều thuộc tính nâng cao. Nhìn chung, tùy qui mô, tùy tính chất của dự án phần mềm mà ta có thể sử dụng các thuộc tính này. Nhưng các thuộc tính nêu trong ví dụ trên là những thuộc tính hay được sử dụng nhất. Để tìm hiểu một cách đầy đủ về các thuộc tính của Hibernate, chúng ta có thể tham khảo tại địa chỉ <http://docs.jboss.org/hibernate/core/3.3/reference/en/html/session-configuration.html>

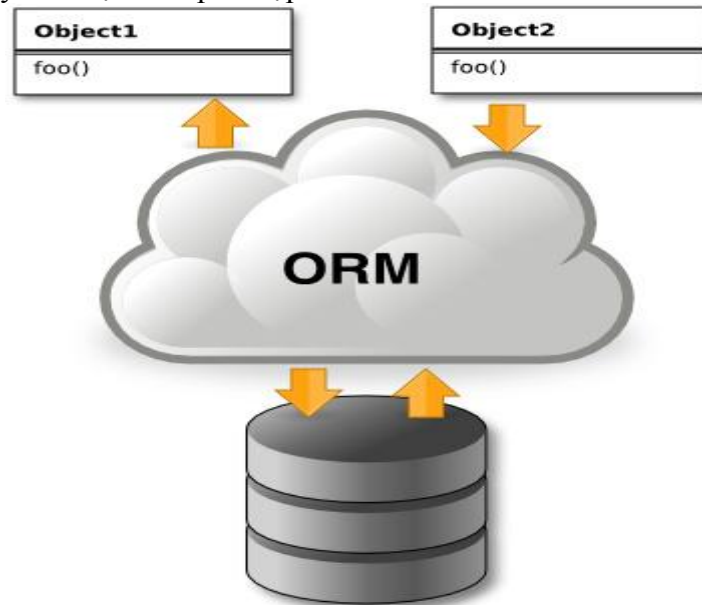
Như vậy, chúng ta đã tìm hiểu xong việc cấu hình kết nối cơ sở dữ liệu trong Hibernate, tiếp sau đây chúng ta sẽ đi vào tìm hiểu chi tiết các kỹ thuật ánh xạ đối tượng – quan hệ, một tính năng then chốt của công nghệ Hibernate.

## **2.Hibernate Object Relational Mapping - ORM :**

Đây là một kỹ thuật cho phép ánh xạ thể hiện của dữ liệu từ các lớp Java đến các bảng dữ liệu tương ứng với những lớp đó. Như vậy, lớp Java là đại diện cho bảng dữ liệu và các thuộc tính của lớp sẽ đại diện cho các cột của bảng.

Người ta gọi những lớp Java này là đối tượng persistence mà ta đã biết định nghĩa về nó ở phần trên. Các lớp này có cấu trúc thông thường giống với Java Bean, đôi khi người ta còn gọi chúng là Plain Old Java Object – POJO. Giữa lớp Java và bảng dữ liệu có mối quan hệ rất chặt chẽ, khi cấu trúc của Java class thay đổi thì đồng nghĩa với việc cấu trúc của các bảng dữ

liệu cũng thay đổi theo, đồng thời khi có sự thay đổi trong cơ sở dữ liệu thì cấu trúc các lớp Java cũng phải thay đổi một cách phù hợp.



Hình 2.3 – ORM

### 2.1)Hibernate XML mapping :

Phương pháp ánh xạ này được thực hiện trong một hay nhiều file XML.

#### a)Ánh xạ đối tượng - quan hệ :

Giả sử ta có một POJO như sau :

```
public class Rank{
    private Long rankid;
    private String name;
    .....
}
```

Để ánh xạ lớp Rank đến bảng rank trong cơ sở dữ liệu ta tạo ra file rank.hbm.xml có nội dung như sau :

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="Rank" table="rank">
    <id name="rankid" type="java.lang.Long">
        <column name="rankid" />
        <generator class="identity" />
    </id>
    <property name="name" type="string">
        <column name="name" length="30" not-null="true" />
    </property>
</class>
</hibernate-mapping>
```

Sau đó thêm vào file hibernate.cfg.xml :

```
<mapping resource="rank.hbm.xml"/>
```

Kết quả thu được như sau :

Name	Type	Length	Decimals	Allow Null	
rankid	bigint	20	0	<input type="checkbox"/>	
name	varchar	30	0	<input type="checkbox"/>	

Hình 2.4 - Bảng rank

**b)Hiểu rõ hơn về ORM :**

Ở ví dụ trên, chúng ta đã sử dụng file rank.hbm.xml để ánh xạ từ đối tượng Rank đến bảng rank trong cơ sở dữ liệu. Bây giờ chúng ta hãy tìm hiểu mỗi thành phần của file ánh xạ này.

□<hibernate-mapping> : Là phần tử gốc của file Hibernate mapping. Giữa thành phần này là các thẻ class.

□<class> : Ánh xạ lớp đối tượng đến thực thể tương ứng trong cơ sở dữ liệu. Nó chỉ ra rằng bảng nào trong cơ sở dữ liệu sẽ được truy cập và cột nào trong bảng sẽ được sử dụng. Mỗi phần tử <hibernate-mapping> có thể có một hay nhiều phần tử <class>

□<id> : Là phần tử định danh duy nhất để định danh một đối tượng. Trên thực tế, phần tử này sẽ ánh xạ đến cột khóa chính trong bảng dữ liệu.

Phần tử này có các thuộc tính sau :

-name : Tên thuộc tính của lớp Java được sử dụng

-column : Tên cột sử dụng để lưu giá trị khóa chính

-type : Kiểu dữ liệu của thuộc tính của lớp Java

-unsaved-value : Là thuộc tính quyết định lớp Java có được khởi tạo hay cập nhật bảng dữ liệu hay không. Nếu giá trị là null thì có nghĩa đối tượng sẽ không ảnh hưởng bất cứ điều gì đến bảng trong cơ sở dữ liệu khi hệ thống khởi động hay shutdown.

□<generator> : Thuộc tính này được sử dụng để sinh khóa chính cho một bộ mới trong bảng dữ liệu. Do cấu trúc của mỗi hệ quản trị cơ sở dữ liệu khác nhau nên Hibernate cung cấp một số giá trị sử dụng để sinh khóa chính như dưới đây.

-increment : Được sử dụng để sinh khóa chính kiểu long, short hay int một cách tự động tăng dần khi không có bất kì tiến trình nào đang thêm dữ liệu vào bảng. Vì vậy, không nên sử dụng giá trị này nếu phần mềm làm việc trong môi trường tập trung.

-identity : Hỗ trợ làm cột khóa chính cho DB2, MySQL, MS SQL Server, Sysbase và HypersonicSQL. Trả về giá trị kiểu long, short hay int.

-sequence : Hibernate có khả năng sử dụng tuần tự để sinh khóa chính. Giá trị này thường được sử dụng khi làm việc với Oracle, Postgresql, DB2.

-assigned : Khi sử dụng giá trị này, mỗi lần thêm vào bảng bộ mới, chúng ta phải chỉ định giá trị cho khóa chính một cách tường minh.

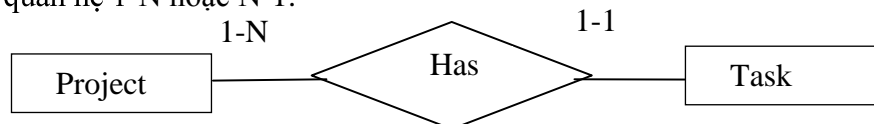
-native : Cho phép chọn identity hay sequence một cách linh động tùy thuộc vào hệ quản trị cơ sở dữ liệu bên dưới.

□<property> : Là phần tử định nghĩa cho các thuộc tính Java ( trừ thuộc tính khóa chính ) và ánh xạ chúng đến các cột trong bảng dữ liệu.

**c)Biểu diễn các mối quan hệ :**

Ở trên là minh họa cách ánh xạ một lớp Java đến một bảng trong cơ sở dữ liệu một cách đơn giản nhất. Nhưng trên thực tế, giữa các bảng thường có mối quan hệ lẫn nhau. Vậy làm sao để biểu diễn các mối quan hệ đó? Hibernate đã cung cấp một cách đầy đủ cho chúng ta thực hiện công việc này. Sau đây là cách biểu diễn các mối quan hệ bằng XML.

□Mối quan hệ 1-N hoặc N-1:



```

public class Project{
    private Long projectid;
    private String name;
    private Set<Task> tasks = new HashSet<Task>(0);
    .....
}
  
```


```

<class name="Project" table="project">
    <id name="projectid" type="java.lang.Long">
        <column name="projectid" />
        <generator class="identity" />
  
```

```

</id>
<property name="name" type="string">
    <column name="name" length="100" not-null="true" />
</property>
<set name="tasks" table="task">
    <key>
        <column name="projectid" not-null="true" />
    </key>
    <one-to-many class="Task" />
</set>
</class>

```

Name	Type	Length	Decimals	Allow Null	
projectid	bigint	20	0	<input type="checkbox"/>	
name	varchar	100	0	<input type="checkbox"/>	

Hình 2.5 - Bảng project

```


public class Task{
    private Long taskid;
    private String name;
    private Project project;
    .....
}

```

```

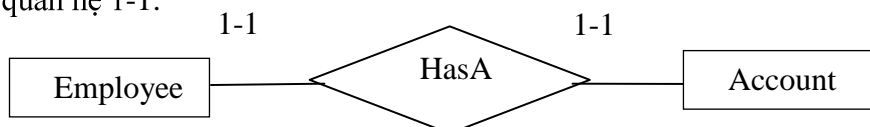
<class name="Task" table="task">
    <id name="taskid" type="java.lang.Long">
        <column name="taskid" />
        <generator class="identity" />
    </id>
    <property name="name" type="string">
        <column name="name" length="100" not-null="true" />
    </property>
    <many-to-one name="project" class="Project">
        <column name="projectid" not-null="true" />
    </many-to-one>
</class>

```

Name	Type	Length	Decimals	Allow Null	
taskid	bigint	20	0	<input type="checkbox"/>	
name	varchar	100	0	<input type="checkbox"/>	
projectid	bigint	20	0	<input type="checkbox"/>	

Hình 2.6 - Bảng task

□Mối quan hệ 1-1:



```

public class Employee{
    private Long employeeid;
    private String name;
    private Account account;
    .....
}

```

```


<class name="Employee" table="employee">
    <id name="employeeid" type="java.lang.Long">
        <column name="employeeid" />

```

```

        <generator class="identity" />
    </id>
    <property name="name" type="string">
        <column name="name" length="100" not-null="true" />
    </property>
    <one-to-one name="account" class="Account" />
</class>

```


Name	Type	Length	Decimals	Allow Null	
employeeid	bigint	20	0	<input type="checkbox"/>	
name	varchar	100	0	<input type="checkbox"/>	

Hình 2.7 - Bảng employee

```

public class Account{
    private Long accountid;
    private Employee employee;
    private String loginname;
    private String password;
    .....
}
<class name="Account" table="account">
    <id name="accountid" type="java.lang.Long">
        <column name="accountid" />
        <generator class="identity" />
    </id>
    <many-to-one name="employee" class="Employee" >
        <column name="employeeid" not-null="true" unique="true" />
    </many-to-one>
    <property name="loginname" type="string">
        <column name="loginname" length="100" not-null="true" />
    </property>
    <property name="password" type="string">
        <column name="password" length="100" not-null="true" />
    </property>
</class>

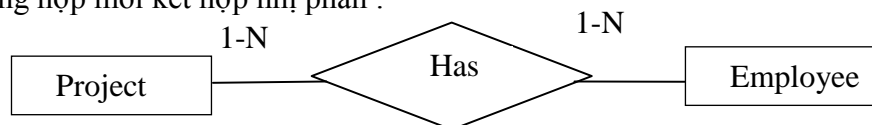
```

Name	Type	Length	Decimals	Allow Null	
accountid	bigint	20	0	<input type="checkbox"/>	
employeeid	bigint	20	0	<input type="checkbox"/>	
loginname	varchar	100	0	<input type="checkbox"/>	
password	varchar	100	0	<input type="checkbox"/>	

Hình 2.8 - Bảng account

□ Mối quan hệ N-N :

Trường hợp mối kết hợp nhị phân :



```

public class Project{
    private Long projectid;
    private String name;
    private Set<Employee> employees = new HashSet<Employee>(0);
    .....
}



```



```

<class name="Project" table="project">
  <id name="projectid" type="java.lang.Long">
    <column name="projectid" />
    <generator class="identity" />
  </id>
  <property name="name" type="string">
    <column name="name" length="100" not-null="true" />
  </property>
  <set name="employees" table="project_employee">
    <key>
      <column name="projectid" not-null="true" />
    </key>
    <many-to-many class="Employee" />
  </set>
</class>
public class Employee{
  private Long employeeid;
  private String name;
  private Set<Project> projects = new HashSet<Project>(0);
  .....
}
<class name="Employee" table="employee">
  <id name="employeeid" type="java.lang.Long">
    <column name="employeeid" />
    <generator class="identity" />
  </id>
  <property name="name" type="string">
    <column name="name" length="100" not-null="true" />
  </property>
  <set name="projects" table="project_employee">
    <key>
      <column name="employeeid" not-null="true" />
    </key>
    <many-to-many class="Project" />
  </set>
</class>

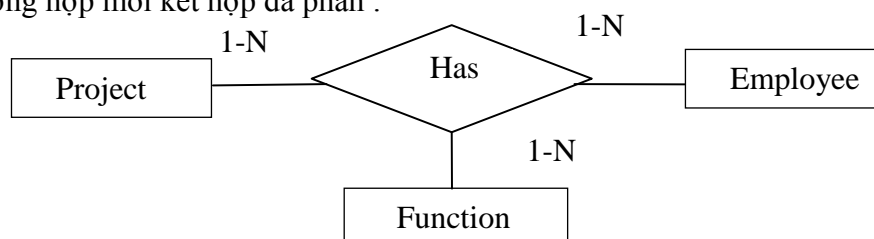
```

Name	Type	Length	Decimals	Allow Null	
projectid	bigint	20	0	<input type="checkbox"/>	
employeeid	bigint	20	0	<input type="checkbox"/>	

Hình 2.9 - Bảng kết hợp của project & employee

Kết quả của mối quan hệ trên là một bảng thứ 3 có khóa chính bao gồm 2 khóa chính của bảng project và bảng employee kết hợp lại.

Trường hợp mỗi kết hợp đa phân :






```

public class EmployeeFunctionInProject{
    private EmployeeFunctionInProjectId id;
    private Employee employee;
    private Project project;
    private Function function;
    .....
}
public class EmployeeFunctionInProjectId{
    private long functionid;
    private long projectid;
    private long employeeid;
    .....
}
<class name="EmployeeFunctionInProject" table="employeefunctioninproject">
    <composite-id name="id" class="EmployeeFunctionInProjectId">
        <key-property name="functionid" type="long">
            <column name="functionid" />
        </key-property>
        <key-property name="projectid" type="long">
            <column name="projectid" />
        </key-property>
        <key-property name="employeeid" type="long">
            <column name="employeeid" />
        </key-property>
    </composite-id>
    <many-to-one name="employee" class="Employee">
        <column name="employeeid" not-null="true" />
    </many-to-one>
    <many-to-one name="project" class="Project">
        <column name="projectid" not-null="true" />
    </many-to-one>
    <many-to-one name="function" class="Function">
        <column name="functionid" not-null="true" />
    </many-to-one>
</class>

```

Tương tự ta thu được :

Name	Type	Length	Decimals	Allow Null	
projectid	bigint	20	0	<input type="checkbox"/>	
employeeid	bigint	20	0	<input type="checkbox"/>	
functionid	bigint	20	0	<input type="checkbox"/>	

Hình 2.10 - Bảng *employeefunctioninproject*

Chúng ta vừa tìm hiểu kỹ thuật ánh xạ đối tượng – quan hệ bằng phương pháp sử dụng XML. Tiếp sau đây sẽ là nội dung của phương pháp ánh xạ bằng cách sử dụng các annotations.

## 2.2) Hibernate Annotations Mapping:

### a) Ánh xạ đối tượng - quan hệ :

Các annotations luôn bắt đầu bằng `@Entity`, đây là dấu hiệu để Hibernate phân biệt các lớp persistence với các lớp bình thường khác. Sau đây là một số annotation thường được sử dụng.

-`@Entity` : Khai báo một lớp Java được xem như là một đối tượng persistence

-`@Id` : Khai báo thuộc tính xác định cho lớp, thuộc tính này sẽ là khóa chính trong bảng dữ liệu.

-@GeneratedValue : Định nghĩa các kiểu sinh giá trị cho khóa chính. Có 4 kiểu chính AUTO, IDENTITY, SEQUENCE, TABLE.

AUTO : Cho phép lựa chọn một cách linh hoạt IDENTITY, SEQUENCE, TABLE để phù hợp với các hệ quản trị cơ sở dữ liệu.

-@Table : Khai báo bảng dữ liệu

-@Column : Khai báo cột trong bảng dữ liệu

-@Version : Khai báo một cột đặc biệt trong bảng dữ liệu. Cột này có tính chất mỗi lần có một hành động làm thay đổi dữ liệu trên bộ chứa nó thì giá trị của nó sẽ tăng lên 1 đơn vị. Ý nghĩa của nó là làm tránh việc cập nhật dữ liệu đồng thời (concurrent update) trên dữ liệu. Khi bắt đầu thực hiện giao dịch, giá trị cột version của bộ sẽ được đọc và lưu lại trong bộ nhớ. Trường hợp khi có nhiều giao dịch đang xảy ra trên dữ liệu, một giao dịch này làm cho 1 bộ nào đó trong bảng bị thay đổi, khi đó version của bộ đó sẽ tự động tăng lên 1 đơn vị, nếu một giao dịch khác muốn cập nhật record đó thì sẽ không thành công. Bởi vì việc cập nhật dữ liệu chỉ cho phép khi version lưu trong bộ nhớ và trong cơ sở dữ liệu phải bằng nhau. Kỹ thuật này còn được gọi với cái tên khác là Optimistic Locking. Trái ngược với Optimistic Locking là Pessimistic Locking, kỹ thuật này không cho phép một giao dịch được thực hiện khi đang có một giao dịch khác đang thực hiện. Cho đến khi giao dịch hiện tại thực hiện hoàn tất thì các giao dịch khác mới thực hiện được.

-@Transient : Chỉ ra rằng thuộc tính không phải là persistence. Nếu muốn một thuộc tính nào của lớp Java sẽ không xuất hiện trong bảng của cơ sở dữ liệu thì phải đặt @Transient trước thuộc tính đó.

- @UniqueConstraint : Ràng buộc unique

-@CheckConstraint : Ràng buộc check

-@OneToOne : Dùng để liên kết các thực thể sử dụng mối quan hệ 1-1

-@OneToMany : Dùng để liên kết các thực thể sử dụng mối quan hệ 1-N

-@ManyToOne : Dùng để liên kết các thực thể sử dụng mối quan hệ N-1

-@ManyToMany : Dùng để liên kết các thực thể sử dụng mối quan hệ N-N

-@JoinColumn : Chỉ định cột tham gia vào mối kết hợp

-@JoinTable : Chỉ định bảng kết quả của mối kết hợp giữa các thực thể

Để minh họa cách ánh xạ, chúng ta sẽ sử dụng lại các lớp POJO ở phần trên.

@Entity

@Table(name = "rank")

```
public class Rank{
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    @Column(name = "rankid", nullable = false, updatable = false)
```

```
    private Long rankid;
```

```
    @Column(name = "name", length = 100, nullable = false)
```

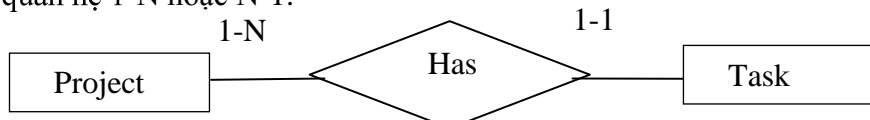
```
    private String name;
```

```
    .....
```

```
}
```

## b) Biểu diễn các mối quan hệ :

□ Mối quan hệ 1-N hoặc N-1:



@Entity

@Table(uniqueConstraints = {

@UniqueConstraint(columnNames = {"number", ..... })))

```
public class Project{
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```

        @Column(nullable = false, updatable = false)
        private Long projectid;
        @Column(nullable = false)
        private int number;
        @Column(length =100 , nullable = false)
        private String name;
        @OneToMany(mappedBy="project")
        private Set<Task> tasks = new HashSet<Task>(0);
        .....
    }

```

Ta sử dụng @UniqueConstraint với cột number để mỗi dự án sẽ có một mã số duy nhất. Thuộc tính mappedBy để chỉ ra đối tượng nào là owner của mối quan hệ.

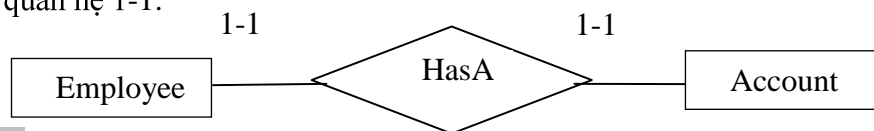
@Entity

```

public class Task{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column( nullable = false, updatable = false)
    private Long taskid;
    @Column(length =100 , nullable = false)
    private String name;
    @ManyToOne
    @JoinColumn(name = "projectid", nullable = false)
    private Project project;
    .....
}

```

□ Mối quan hệ 1-1:



@Entity

```

public class Employee{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column( nullable = false, updatable = false)
    private Long employeeid;
    @Column(length =100 , nullable = false)
    private String name;
    @OneToOne(mappedBy="employee")
    private Account account;
    .....
}

```

@Entity

```

public class Account{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column( nullable = false, updatable = false)
    private Long accountid;
    @OneToOne
    @JoinColumn(name="employeeid",unique=true)
    private Employee employee;
    @Column(length =100 , nullable = false)
    private String loginname;
}

```

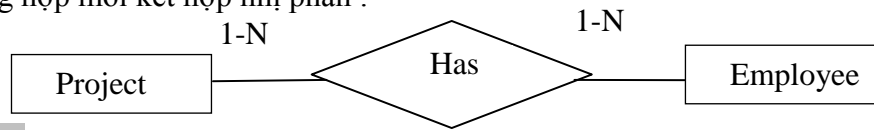
```

        @Column(length = 100 , nullable = false)
        private String password;
        .....
    }

```

□Mối quan hệ N-N :

Trường hợp mỗi kết hợp nhị phân :



@Entity

```

public class Project{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column( nullable = false, updatable = false)
    private Long projectid;
    @Column(length = 100 , nullable = false)
    private String name;
    @ManyToMany
    @JoinTable(name = "projectemployee",
        joinColumns = { @JoinColumn(name = "projectid") },
        inverseJoinColumns = { @JoinColumn(name = "employeeid") })
    private Set<Employee> employees = new HashSet<Employee>(0);
    .....
}

```

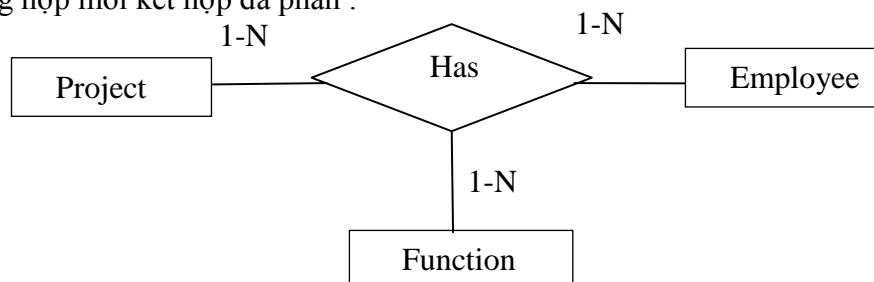
@Entity

```

public class Employee{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column( nullable = false, updatable = false)
    private Long employeeid;
    @Column(length = 100 , nullable = false)
    private String name;
    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "projectemployee",
        joinColumns = { @JoinColumn(name = "employeeid") },
        inverseJoinColumns = { @JoinColumn(name = "projectid") })
    private Set<Project> projects = new HashSet<Project>(0);
    .....
}

```

Trường hợp mỗi kết hợp đa phân :



```

public class EmployeeFunctionInProjectId{
    private long functionid;
    private long projectid;
    private long employeeid;
}

```

```

.....
}
@Entity
@IdClass(EmployeeFunctionInProjectId.class)
public class EmployeeFunctionInProject{
    @EmbeddedId
    private EmployeeFunctionInProjectId id;
    @ManyToOne
    @JoinColumn(name = "employeeid", nullable = false)
    private Employee employee;
    @ManyToOne
    @JoinColumn(name = "projectid", nullable = false)
    private Project project;
    @ManyToOne
    @JoinColumn(name = "functionid", nullable = false)
    private Function function;
    .....
}

```

Tất cả kết quả mà chúng ta thu được đều giống như trường hợp sử dụng XML.

Như vậy, đến đây chúng ta đã tìm hiểu qua kỹ thuật ORM trong Hibernate. Trong phần tiếp theo, chúng ta sẽ tìm hiểu cách Hibernate làm việc với cơ sở dữ liệu thông qua ngôn ngữ truy vấn HQL và ngôn ngữ truy vấn theo tiêu chuẩn.

### III.Hibernate Query Language - HQL:

#### 1.Định nghĩa :

Đây là ngôn ngữ truy vấn dữ liệu mà Hibernate sử dụng để thực hiện các giao dịch của người sử dụng. Về cú pháp thì phần lớn rất giống SQL, nhưng điểm khác biệt là HQL sử dụng lớp và thuộc tính thay cho bảng và tên cột. Vì vậy , HQL cung cấp tính đa hình, kế thừa và ít rườm rà hơn SQL.

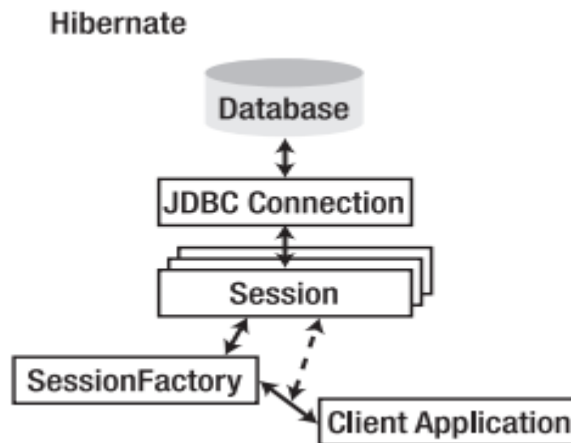
Những ưu điểm của HQL :

- Cho phép thể hiện các câu truy vấn SQL dưới dạng các đối tượng.
- Thay vì trả về kết quả dưới dạng thô, câu truy vấn HQL trả về kết quả truy vấn dưới dạng đối tượng hoặc bộ đối tượng để ứng dụng dễ dàng sử dụng và thao tác các hành động trên các đối tượng này.
- Cung cấp đầy đủ các truy vấn đa hình. Điều đó có nghĩa là cùng với đối tượng được trả về trong kết quả truy vấn thì tất cả đối tượng có liên quan đến đối tượng cũng sẽ được trả về.
- Là ngôn ngữ dễ học và dễ thực thi trong các ứng dụng bởi vì cú pháp và tính chất rất giống SQL.
- Cung cấp đầy đủ các tính năng giống như SQL: inner join, outer join, các phép chiếu, phép kết, các hàm tính gộp như avg,max,sum...
- Dễ dàng tạo ra các truy vấn độc lập sử dụng kiểu dữ liệu hoàn toàn độc lập với kiểu dữ liệu trong hệ quản trị cơ sở dữ liệu bởi vì cung cấp cơ chế chuyển đổi kiểu dữ liệu ngôn ngữ lập trình sang kiểu dữ liệu database trong quá trình thực thi.

Nhược điểm : tốc độ thực thi chậm hơn SQL vì phải trải qua giai đoạn chuyển đổi từ HQL sang SQL.

#### 2.Thực hiện truy vấn cơ sở dữ liệu :

Hibernate thực hiện các giao dịch bằng một đối tượng đặc biệt đó là đối tượng Session. Vòng đời của Session này bắt đầu từ lúc bắt đầu giao dịch và kết thúc khi giao dịch kết thúc. Đối tượng Session này được tạo ra từ đối tượng SessionFactory .



Hình 2.11 - Mô hình kết nối trong Hibernate

Chức năng cụ thể của Session là khởi tạo, đọc, xóa ,cập nhật các thực thể ánh xạ bởi lớp Java trong cơ sở dữ liệu.

**a) Các bước thực hiện một giao dịch trong HQL :**

□ Bước 1 : Tạo đối tượng SessionFactory

+ Trường hợp sử dụng XML mapping

```
SessionFactory sf = new Configuration().configure().buildSessionFactory();
```

+ Trường hợp sử dụng annotation

```
SessionFactory sf = new
AnnotationConfiguration().configure().buildSessionFactory();
```

Chú ý : Cả 2 câu lệnh trên sẽ tự động tìm và đọc file hibernate.cfg.xml để thực thi

□ Bước 2 : Tạo đối tượng Session

```
Session session = null;
```

□ Bước 3 : Mở session

```
session = sf.openSession();
```

□ Bước 4 : Mở giao dịch

```
Transaction tr = sess.beginTransaction();
```

□ Bước 5 : Thực hiện các thao tác

```
String sql = ...
```

```
Query query = session.createQuery(sql);
```

```
List results = query.list();
```

Như vậy kết quả trả về của tất cả các câu truy vấn là một danh sách.

□ Bước 6 : Kết thúc giao dịch

```
tr.commit();
```

□ Bước 7 : Đóng session

```
session.flush();
```

```
session.close();
```

**b) Cách viết câu truy vấn HQL :**

● Mệnh đề FROM :

Cú pháp : from *object* [as *object\_alias*]

Ví dụ : from Project as p hoặc from Project p

Câu truy vấn trên sẽ lấy toàn bộ các record có trong bảng project.

● Mệnh đề SELECT :

Cú pháp : select [*object.*] *property*

Ví dụ : select p.name from Project p

● Mệnh đề WHERE :

Cú pháp : where *condition*

Ví dụ : select p from Project p where p.status.name = 'finished'

Câu truy vấn sẽ lấy ra danh sách những dự án có trạng thái là finished

●Mệnh đề INSERT ,UPDATE, DELETE:

Đối với việc thêm, xóa, sửa dữ liệu, thay vì phải viết câu lệnh như SQL thì Hibernate đã hỗ trợ sẵn một số phương thức trong session để thực hiện công việc này.

Cú pháp :

```
session.save(object);
session.saveOrUpdate(object);
session.update(object);
session.delete(object);
```

Ví dụ :

```
Project project= new Project();
project.setNumber(100);
project.setName("Helios");
project.setCustomer("ABC");
project.setStatus("Finished");
project.setStartDate(new Date("1/1/2010"));
project.setEndDate(new Date("1/10/2011"));
session.save(project);
Task task = session.get(Task.class,taskId);
task.setStatus("resolved");
session.update(task);
User user = session.get(User.class,userId);
session.delete(user);
```

●Mệnh đề ORDER BY :

Cú pháp : order by *object0.property0* [asc|desc][,*object1.property0*]...

●Mệnh đề GROUP BY :

Cú pháp : group by *object0.property0*[,*object1.property0*]...

●Các mệnh đề tính gộp :

```
count( [ distinct | all ] object | object.property )
count(*)
sum ( [ distinct | all ] object.property)
avg( [ distinct | all ] object.property)
max( [ distinct | all ] object.property)
min( [ distinct | all ] object.property)
```

●Phép kết trong Hibernate : Hibernate cung cấp 4 loại kết sau

- inner join
- left outer join
- right outer join
- full join (ít được sử dụng trong thực tế)

Ví dụ : select project

```
from Project as project inner join project.customer as customer
where customer.name='ABC'
```

Câu truy vấn trên sẽ lấy danh sách các dự án mà khách hàng tên là ABC.

Tuy nhiên, HQL tỏ ra không hiệu quả trong các truy vấn đa tiêu chuẩn - Multi Criteria Search. Để khắc phục, Hibernate cung cấp thêm một phương pháp truy vấn khác, phương pháp truy vấn sử dụng các hàm thư viện Hibernate Criterial Query API. Chúng ta sẽ tìm hiểu phương pháp này ngay trong phần tiếp theo.



#### IV. Hibernate Criteria Query :

##### 1. Định nghĩa :

Criteria Query cho phép tạo và thực thi các câu truy vấn động hướng đối tượng. Nó có thể thay thế cho HQL trong các câu truy vấn đa tiêu chuẩn. Khi mà HQL không thực sự tỏ ra có hiệu quả trong các loại câu truy vấn này.

##### 2. Cách viết câu truy vấn :

Các bước viết câu truy vấn khi sử dụng Hibernate Criteria Query cũng giống như các bước viết câu truy vấn HQL. Sau đây là bước thực hiện các thao tác với Hibernate Criteria Query.

Ví dụ :

```
Criteria crit = session.createCriteria(Project.class);
```

```
List projects = crit.list();
```

Câu truy vấn trên sẽ lấy toàn bộ các record có trong bảng project và trả về một danh sách.

Interface Criteria còn hỗ trợ các phương thức cho việc thực hiện các tính năng liên quan đến việc phân trang (pagination). Trong đó chú ý đến 2 phương thức bên dưới đây :

```
crit.setFirstResult(int arg0); //Chỉ định vị trí record bắt đầu sẽ được đưa vào kết quả.
```

```
crit.setMaxResults(int arg0); //Chỉ định số lượng tối đa các record sẽ có mặt trong kết quả.
```

Ngoài ra còn cung cấp distinct dữ liệu thu được :

```
crit.uniqueResult();
```

Để giới hạn kết quả trả về của câu truy vấn, interface Criteria được hỗ trợ thêm phương thức add với tham số là điều kiện trích rút.

Cú pháp : add(Criterion arg0)

Như vậy, mỗi điều kiện được xem là một đối tượng Criterion.

□ Lớp Restrictions : Cung cấp các phương thức static làm nhiệm vụ trích rút dữ liệu theo một hay nhiều điều kiện nào đó.

Ví dụ :

```
DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
```

```
Criteria crit= session.createCriteria(Project.class);
```

```
crit.add(Restrictions.like("name", "%abc%"))
```

```
.createAlias("customer", "customer")
```

```
.add(Restrictions.eq("customer.name", "xyz"))
```

```
.add(Restrictions.between("startdate", df.parse("2011-1-1"), df.parse("2011-12-12")))
```

```
.....;
```

Câu truy vấn trên sẽ lấy ra các dự án có chuỗi ký tự “abc” trong tên dự án, có khách hàng tên là “xyz”, có ngày bắt đầu nằm trong khoảng từ ngày 2011-1-1 đến ngày 2011-12-12....

Không những thế, lớp Restrictions còn hỗ trợ nhiều phương thức khác, bao gồm nhiều phép toán so sánh (>, >=, <, <=, !=), phép toán logic(AND, OR, NOT), IN, ...

□ Sắp xếp kết quả truy vấn : Được thực hiện thông qua phương thức addOrder(Order).

```
Criteria crit= session.createCriteria(Project.class);
```

```
crit.add(Restrictions.gt("number", 100).addOrder(Order.asc("name"))).
```

```
addOrder(Order.desc("number"));
```

Câu truy vấn trên sẽ lấy ra danh sách tất cả các dự án có number>100 (gt : viết tắt của greater than), sau đó sắp xếp kết quả tăng dần theo tên và giảm dần theo number.

□ Phép chiếu :

Ví dụ : Để lấy cột name và number trong bảng Project ta làm như sau

```
Criteria crit= session.createCriteria(Project.class);
```

```
ProjectionList pl = Projections.projectionList();
```

```
pl.add(Projections.property("name"));
```

```
pl.add(Projections.property("number"));
```

```
crit.setProjection(pl);
```

```
List results = crit.list();
```

Trong đó, mỗi phần tử của danh sách là một mảng các đối tượng ( Object[] ).

Bên cạnh hỗ trợ việc thực hiện phép chiếu, Hibernate còn tích hợp cả các hàm thống kê (min, max, sum, avg, count...) vào trong lớp Projections.

Cú pháp : Projections.min|max|sum|avg|count...( propertyName)

Ví dụ :

```
Criteria crit= session.createCriteria(Project.class);
crit.setProjection(Projections.max("number"));
```

Câu truy vấn trên sẽ lấy ra dự án có number lớn nhất.

□ Phép kết : Criteria thực hiện phép kết thông qua phương thức createAlias()

```
Criteria crit= session.createCriteria(Project.class);
crit.createAlias("customer", "customer")
      .add(Restrictions.eq("customer.name", "abc"))
      .createAlias("group", "group")
      .createAlias("group.leader", "leader")
      .add(Restrictions.eq("leader.visa", "xyz"));
```

Trong đó customer là một thuộc tính kiểu đối tượng Customer của lớp Project.

Câu truy vấn trên sẽ trả về danh sách những dự án có khách hàng có tên là xyz , có group leader với visa là xyz.

□ Gom nhóm kết quả truy vấn :

```
Criteria crit= session.createCriteria(Project.class);
crit.setProjection( Projections.groupProperty("customer"))
      .addOrder( Order.asc("number") );
```

Câu truy vấn trên sẽ trả về danh sách những dự án được gom nhóm theo khách hàng và được sắp xếp theo chiều tăng dần của number.

Đến đây, chúng ta đã tìm hiểu xong những vấn đề cơ bản của Hibernate framework. Trong phần tiếp theo, chúng ta sẽ tiếp tục nghiên cứu một trong những framework nổi tiếng đồ sộ trong Java nữa đó là Spring MVC framework. Có thể nói sự kết hợp giữa Hibernate và Spring MVC trong các dự án phần mềm là sự kết hợp tuyệt vời. Nếu như Hibernate là công cụ quản lý cơ sở dữ liệu thì Spring MVC lại là công cụ sử dụng kết quả của các truy vấn đến dữ liệu đó .

**CHƯƠNG III****SPRING MVC FRAMEWORK****I. Tổng quan về Spring :****1. Định nghĩa :**

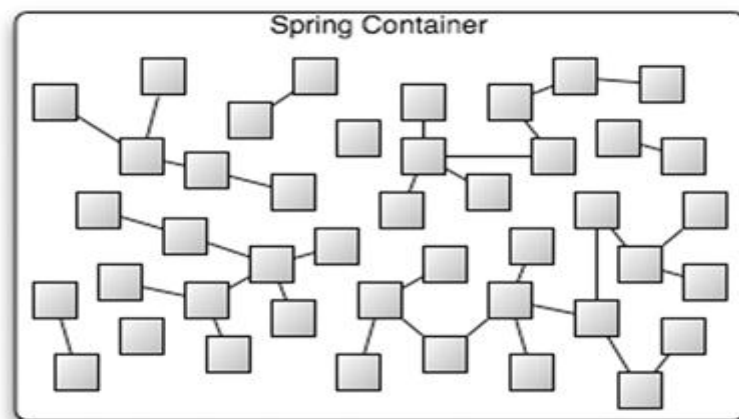
Spring là một open source framework được tạo ra bởi Rod Johnson và được thể hiện trong cuốn sách “Expert One-on-One: J2EE Design and Development” của chính tác giả. Spring được sinh ra nhằm hướng vào việc phát triển các ứng dụng thương mại phức tạp. Spring có thể làm rất nhiều thứ bởi đây là một framework rất đồ sộ, nhưng nếu trải Spring thành từng phần cơ bản thì Spring là một lightweight, dependency injection, aspect-oriented container và framework.

◇ Lightweight : kích thước của Spring có thể được tập trung trong các file jar có kích thước chỉ khoảng 2.5MB.

◇ Dependency injection : Spring đề xướng “loose coupling” tức là sự ghép nối lỏng giữa các đối tượng thông qua một kỹ thuật gọi là Dependency Injection (DI). Thuật ngữ “loose coupling” có nghĩa rằng các đối tượng không thật sự cần biết về sự định nghĩa của các đối tượng khác mặc dù giữa chúng có mối liên quan với nhau. Khi áp dụng kỹ thuật này, các đối tượng được sẽ truyền (hay tiêm) sự phụ thuộc thay vì phải tạo ra và tìm kiếm các đối tượng phụ thuộc cho chúng. Để làm việc này, Spring đã xây dựng một đối tượng Spring Container, chính đối tượng này sẽ đưa sự phụ thuộc vào đối tượng khi mà đối tượng đó được sinh ra mà không cần phải chờ được yêu cầu.

◇ Aspect-oriented : Spring mang đến sự hỗ trợ đầy đủ cho Aspect-oriented Programing – AOP. Đây là một cách lập trình mới, xuất hiện sau lập trình OOP. Trong kiểu lập trình này cho phép chúng ta thực hiện các vấn đề riêng biệt một cách linh hoạt và kết hợp chúng lại để tạo nên hệ thống sau cùng. Phương pháp này tạo điều kiện thuận lợi cho bảo trì và kiểm tra phần mềm.

◇ Container : Spring chứa đựng, quản lý vòng đời và cấu hình các đối tượng của ứng dụng. Trong Spring, chúng ta có thể nắm rõ các đối tượng của ứng dụng được tạo ra như thế nào, các đối tượng đó được cấu hình ra sao và mối quan hệ giữa các đối tượng chúng.



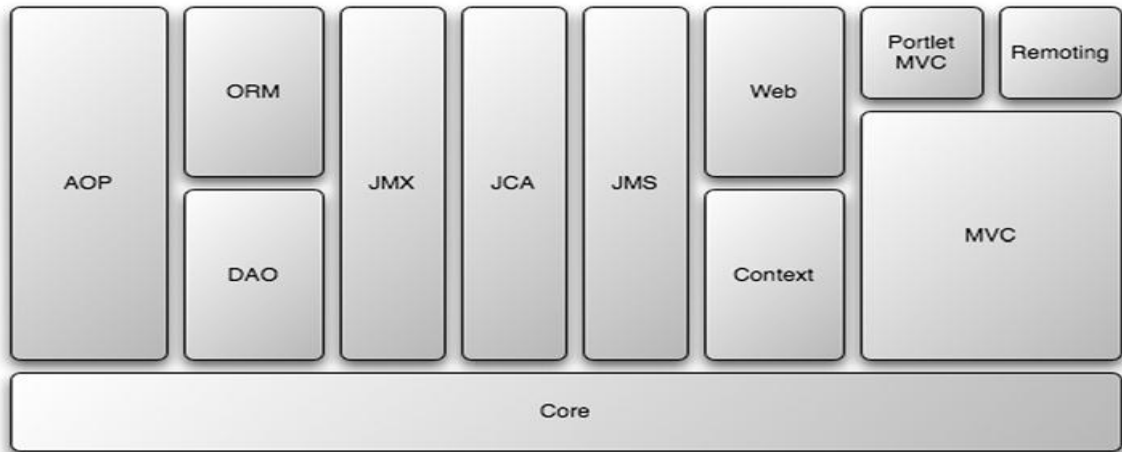
Hình 3.1 - Spring Container

Trong các ứng dụng dựa trên Spring. Các đối tượng của ứng dụng được chứa trong Spring container. Container tạo ra các đối tượng, liên kết chúng lại với nhau ( chính là DI ), cấu hình và quản lý vòng đời của chúng.

◇ Framework : Spring có tạo ra những ứng dụng phức tạp bằng các thành phần đơn giản. Trong Spring, các đối tượng thường được khai báo trong file XML. Spring cũng cung cấp nhiều chức năng cơ sở hạ tầng như quản lý giao dịch, khả năng tích hợp với các framework khác.

## 2.Kiến trúc Spring :

Dưới đây là các module của Spring :



Hình 3.2 - Kiến trúc Spring framework

- Core : Là module cơ bản của Spring. Module này cung cấp các chức năng chủ yếu của Spring framework. Trong module này chứa đựng BeanFactory, đây là container quan trọng của Spring, và là nguồn gốc của kỹ thuật DI.
- Application context: Module này xây dựng trên container core. Nếu BeanFactory thể hiện tính container của Spring thì Application context lại thể hiện tính framework. Không những kế thừa từ BeanFactory, module này còn được trang bị thêm nhiều tính năng như multilanguagel (đa ngôn ngữ), chức năng xác thực cho ứng dụng.
- AOP : Cung cấp đầy đủ cho Aspect-oriented programming
- DAO : Viết tắt của data access object ,cung cấp các mã lệnh theo khuôn dạng để tạo kết nối, câu lệnh, xử lý kết quả truy vấn, và sau đó đóng các kết nối.
- ORM(Object-relational mapping) : Cho phép chuyển đổi dữ liệu giữa các kiểu hệ thống khác nhau. Kỹ thuật này cho phép tạo ra một đối tượng cơ sở dữ liệu ảo có thể sử dụng trong các ngôn ngữ lập trình.
- JMX(Java Management Extensions) : Module này thể hiện toàn bộ quá trình hoạt động ở bên trong ứng dụng, điều đó làm cho chương trình được quản lý và cấu hình lại một cách dễ dàng.
- JCA(Java EE Connector API) : Cung cấp kỹ thuật cho phép hợp nhất các ứng dụng Java từ nhiều hệ thống thông tin, nhiều cơ sở dữ liệu và server khác nhau.
- Spring MVC : Cung cấp phương pháp xây dựng các ứng dụng theo mô hình MVC
- Portlet MVC : Các ứng dụng web thường dựa trên nền là các trang web, mỗi yêu cầu đến ứng dụng sẽ được hiển thị kết quả lên một trang mới. Portlet MVC cho phép thể hiện nhiều chức năng của ứng dụng chỉ trên một trang web (single web page).
- Web: Cung cấp các class đặc biệt cho Spring MVC và Portlet MVC để cho phép upload những multipart file . Ngoài ra, module này còn được tích hợp nhiều cung cấp cho các framework opensource khác
- Remoting: Để các ứng dụng khác nhau có thể liên kết với nhau trong quá trình hoạt động . Đặc biệt khi các ứng dụng truy cập mạng , các Remoting sẽ được sử dụng để làm nhiệm vụ truyền.
- JMS (Java Message Service): Cung cấp các chức năng cho dịch vụ tin nhắn ,trong đó tin nhắn sẽ được đưa vào hàng đợi để gửi đi nhằm tăng tốc độ thực thi.

Các module này có thể cung cấp cho chúng ta rất nhiều chức năng để xây dựng bất kì một ứng dụng thương mại nào. Nhưng không vì thế mà chúng ta dựa tất cả vào Spring. Chúng ta hoàn toàn có thể tự do lựa chọn những module phù hợp với ứng dụng và tìm kiếm những hỗ trợ khác, những hỗ trợ mà Spring không đáp ứng đủ yêu cầu.

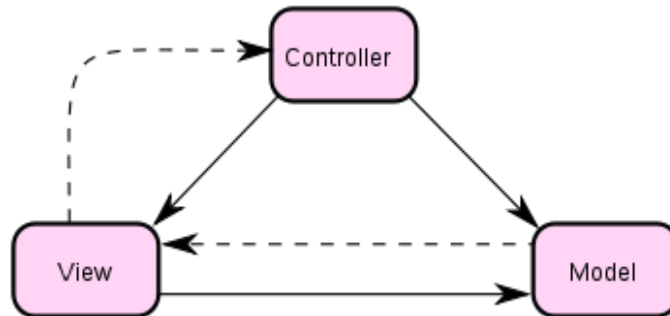
Trong khuôn khổ của bài báo cáo này, chúng ta sẽ cùng nhau nghiên cứu về MVC module.

## II. Giới thiệu Spring MVC framework :

### 1. Khái quát :

Spring MVC là một trong những module quan trọng của Spring framework. Kế thừa những ưu điểm của Spring và nó đã trở thành một trong những framework nổi tiếng trong phát triển các ứng dụng web.

MVC là viết tắt của Model – View - Controller. Mô hình MVC là một trong những kiến trúc phổ biến trong lĩnh vực công nghệ phần mềm. Mô hình này được minh họa trong hình bên dưới :



Hình 3.3 - Mô hình MVC

+Model : Chịu trách nhiệm quản lý các chức năng và quản lý dữ liệu của ứng dụng, trả lời cho các yêu cầu về thông tin của nó từ View, trả lời cho các yêu cầu thay đổi trạng thái của nó từ Controller .

+View : Hiển thị Model cho user

+Controller : Nhận input từ user và trả lời yêu cầu cho user bằng cách tạo ra các lời gọi đến Model .

### 2. Đặc tính của Spring MVC :

◊Phân cách hoàn toàn các vai trò trong ứng dụng : Việc phân cách các vai trò trong ứng dụng sẽ làm giảm độ phức tạp, chương trình dễ quản lý, từ đó làm tăng độ tin cậy của ứng dụng.

◊Cấu hình đơn giản : Thực hiện trong các file XML.

◊ Pluggable View : Đây là kỹ thuật cho phép Spring MVC có thể làm việc với nhiều công nghệ như JSP, Tiles, Velocity ...

◊ Khả năng sử dụng lại mã nguồn

◊ Dependency Injection : Làm cho tiết kiệm các đoạn code giống nhau và việc viết code sẽ có hiệu quả hơn.

### 3. Thành phần của Spring MVC :

◊Model : Được thể hiện qua lớp ModelAndView. Thường được Spring dùng để nắm giữ dữ liệu . Ngoài ra, Spring còn bao bọc cả các dữ liệu chức năng vào trong class này và truyền chúng đến View.

◊View : Được thể hiện qua lớp ModelAndView.View có thể làm việc với JSP, Tiles, Velocity, Jasper ...

◊Controller: Chịu trách nhiệm thu nhận tất cả các request từ user và xử lý request đó thông qua sự hỗ trợ từ Model.

Để tiếp cận với Spring MVC, chúng ta hãy bắt đầu với ví dụ Hello world sau đây.

### 4. Chương trình Hello World :

Dependency injection – DI là điều cơ bản nhất mà Spring làm. Nhưng như thế nào là DI? Chúng ta sẽ bắt đầu bằng chương trình “Hello World” để biết được Spring làm việc như thế nào.

Lớp đầu tiên là một lớp dịch vụ, đảm nhận trách nhiệm in ra một câu chào hỏi quen thuộc. Lớp này được định nghĩa một cách ngắn gọn thông qua một interface.

```

public interface GreetingService {
    void sayGreeting();
}
  
```

```

public class GreetingServiceImpl implements GreetingService {
    private String greeting;
    public GreetingServiceImpl() {}
    public GreetingServiceImpl(String greeting) {
        this.greeting = greeting;
    }
    public void sayGreeting() {
        System.out.println(greeting);
    }
    public void setGreeting(String greeting) {
        this.greeting = greeting;
    }
}

```

Lớp *GreetingServiceImpl* có duy nhất một thuộc tính là *greeting*. Thuộc tính này đơn giản chỉ là một chuỗi ký tự sẽ giữ nội dung của một thông điệp sẽ được in ra khi ta gọi phương thức *sayGreeting()*. Và chú ý rằng *greeting* có thể được thiết lập bằng 2 cách : constructor hoặc setter method. Chúng ta sẽ để Spring container thiết lập giá trị cho thuộc tính này. File cấu trúc Spring có tên *hello.xml* dưới đây sẽ chỉ ra làm thế nào để cấu hình cho *greeting service* :

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">
    <bean id="greetingService" class="GreetingServiceImpl">
        <property name="greeting" value="Hello World!" />
    </bean>
</beans>

```

File XML này đã khai báo một thực thể của *GreetingServiceImpl* trong Spring container và thiết lập giá trị cho thuộc tính *greeting* là “Hello World!”. Hãy tìm hiểu sâu vào file XML này để biết được chúng làm việc ra sao.

Tại gốc của file cấu trúc XML là phần tử *<beans>*. Nó bao bọc các phần tử *<bean>* và đó cũng là phần tử gốc của bất kỳ file cấu hình Spring nào. Phần tử *<bean>* được sử dụng để cho Spring container biết về một lớp và lớp đó sẽ được cấu hình như thế nào. Ở đây, thuộc tính *id* được sử dụng để đặt tên cho phần tử *bean* và *class* là thuộc tính chỉ định tên lớp đầy đủ của *bean* đó. Trong phạm vi phần tử *<bean>* là phần tử *<property>*, sử dụng để thiết lập giá trị thuộc tính, cụ thể ở đây là thuộc tính *greeting*. Như đã thấy ở trên, phần tử *<property>* này chỉ cho Spring container gọi phương thức *setGreeting()* và truyền giá trị “Hello World!” khi phần tử *bean* này được khởi tạo.

Đoạn trích trên cho thấy rằng Spring container đã làm những gì khi khởi tạo *greeting service* dựa vào định nghĩa XML.

```

GreetingServiceImpl greetingService = new GreetingServiceImpl();
greetingService.setGreeting("Hello World!");

```

Chúng ta cũng hoàn toàn có thể chọn để Spring container thiết lập giá trị cho *greeting* thông qua constructor như sau :

```

<bean id="greetingService" class="GreetingServiceImpl">
    <constructor-arg value=" Hello world!" />
</bean>

```

Đoạn mã sau đây sẽ chỉ ra cách container khởi tạo giá trị cho *greeting service* thông qua phần tử *<constructor-arg>* :

```

GreetingService greetingService = new GreetingServiceImpl("Hello world !");

```

Và cuối cùng trong chương trình Hello World này là một lớp đảm nhận việc load Spring container và sử dụng nó để lấy về kết quả từ greeting service.

```
public class HelloApp {
    public static void main(String[] args) throws Exception {
        BeanFactory factory = new XmlBeanFactory(new FileSystemResource("hello.xml"));
        GreetingService greetingService = (GreetingService)
            factory.getBean("greetingService");
        greetingService.sayGreeting();
    }
}
```

BeanFactory được sử dụng ở đây chính là Spring container. Sau khi load file hello.xml, phương thức main() gọi phương thức getBean() trong BeanFactory để trả về một tham chiếu đến greetingService. Khi đã có greetingService trong tay, cuối cùng gọi sayGreeting. Khi chạy HelloApp, đừng ngạc nhiên tại sao màn hình xuất hiện dòng chữ : Hello World!

Đây chỉ là một chương trình đơn giản để bắt đầu khám phá Spring. Mặc dù nó đơn giản, nhưng nó cũng đã cho thấy rõ những khái niệm cơ bản về cấu hình và sử dụng một lớp trong Spring. Thật không may mắn, nó có lẽ quá đơn giản bởi vì chỉ cho thấy làm sao để cấu hình một bean thông qua việc “tiêm” một string vào một property. Sức mạnh thực sự của Spring nằm trong làm sao để “tiêm” nhiều bean vào một bean khác thông qua DI.

### III.Cơ bản về sự móc nối giữa các bean – Bean wiring:

Như chúng ta đã biết, khắp nơi trong Spring đều là các bean, giữa các bean này móc nối với nhau trong Spring container. Chính Spring container đã tạo ra chúng, quản lý chúng, đồng thời tạo ra mối quan hệ giữa chúng. Không chỉ có BeanFactory, Spring đến với một vài container khác được liệt kê thành hai loại riêng biệt. Các bean thuộc nhóm BeanFactory được định nghĩa bởi interface org.springframework.beans.factory là những container đơn giản nhất, cung cấp hỗ trợ cơ bản cho DI. Các bean thuộc nhóm Application context được định nghĩa bởi org.springframework.context.ApplicationContext xây dựng trên BeanFactory cung cấp các dịch vụ, khả năng chuyển đổi văn bản thông qua các file properties, khả năng tạo ra các sự kiện ứng dụng (Application events) đến máy nghe sự kiện ứng dụng (Event Listeners). Hãy bắt đầu chuyên thám hiểm vào Spring container với container cơ bản nhất : BeanFactory.

#### 1.Giới thiệu về BeanFactory :

Đây là container giữ hoàn toàn trách nhiệm tạo và phân phối các bean. BeanFactory biết có bao nhiêu đối tượng tồn tại trong ứng dụng nên có thể tạo ra sự phối hợp giữa chúng ngay khi chúng được khởi tạo. BeanFactory tham gia vào vòng đời của các bean, gọi các hàm khởi tạo hay hủy bỏ nếu chúng được định nghĩa. Có nhiều sự thực thi của BeanFactory nhưng một trong những lớp được sử dụng phổ biến là org.springframework.beans.factory.xml.XmlBeanFactory , một container sẽ load các bean của nó chứa đựng trong file XML. Để tạo XmlBeanFactory, ta phải truyền một thực thể của org.springframework.core.io.Resource vào phương thức khởi tạo. Đối tượng Resource sẽ cung cấp dữ liệu XML cho factory.

Ví dụ sau là cách sử dụng FileSystemResource để đọc nội dung của file XML từ ổ đĩa hệ thống :

```
BeanFactory factory =
    new XmlBeanFactory(new FileSystemResource("c:/beans.xml"));
```

Đoạn mã lệnh đơn giản trên cho Spring BeanFactory đọc định nghĩa về các bean từ file XML. Nhưng BeanFactory chưa khởi tạo các bean ngay lúc này cho đến khi ứng dụng cần đến chúng.

Để thu về bean từ BeanFactory, đơn giản là chỉ việc gọi phương thức getBean() và truyền vào ID của bean mà ta muốn lấy được.

```
MyBean myBean = (MyBean) factory.getBean("myBean");
```

Khi gọi phương thức getBean(), BeanFactory mới thực sự khởi tạo bean và thiết lập giá trị cho các thuộc tính của bean thông qua DI.

## 2.Làm việc với ApplicationContext :

Một BeanFactory thì làm việc tốt đối với những ứng dụng đơn giản, nhưng để lợi dụng được hết sức mạnh của Spring chúng ta nên load các bean của ứng dụng bằng container cao cấp hơn : ApplicationContext.

Như đã đề cập ở phần trên, ApplicationContext hầu như rất giống với BeanFactory , cả hai đều load định nghĩa về bean, liên kết chúng lại với nhau, phân phát chúng khi có yêu cầu. Nhưng ApplicationContext có thể làm hơn thế nữa :

- Khả năng giải quyết vấn đề đa ngôn ngữ cho ứng dụng thông qua các file properties
- Cung cấp một phương pháp chung để load các file resource, bao gồm cả file image
- Xuất bản các sự kiện(Events) đến các bean thông qua việc đăng ký với các máy nghe(Listeners).

Bởi các tính năng bổ trợ đó, ApplicationContext được sử dụng hầu như ở tất cả các ứng dụng dựa trên Spring. Có lẽ, chỉ có một lần chúng ta có thể xem xét đến việc sử dụng BeanFactory trong hoàn cảnh khi mà resource bị khan hiếm, chẳng hạn đối với các mobile device.

Giữa rất nhiều thực hiện của ApplicationContext, có 3 loại thường được sử dụng :

+ ClassPathXmlApplicationContext : Load định nghĩa về context trong một file XML được chứa đựng trong classpath(thông thường là các file jar)

ApplicationContext context =

new ClassPathXmlApplicationContext("foo.xml");

+ FileSystemXmlApplicationContext : Load định nghĩa về context trong một file XML được chứa đựng trong thiết bị lưu trữ của hệ thống

ApplicationContext context =

new FileSystemXmlApplicationContext("c:/foo.xml");

+XmlWebApplicationContext : Load định nghĩa về context trong một file XML nằm trong phạm vi ứng dụng

ApplicationContext context =

new XmlWebApplicationContext ("foo.xml");

## 3. Làm việc với các Spring Bean:

### 3.1.Tạo một bean đơn giản :

Để cho dễ hiểu hơn về bean, chúng ta hãy đi vào một ví dụ vui sau đây. Chúng ta có một interface Performer đại diện cho những người tham gia vào một cuộc thi biểu diễn.

```
public interface Performer {
    void perform() throws PerformanceException;
}
```

Và sau đây sẽ là thí sinh đầu tiên đến từ Juggler được mang tên Duke.

```
public class Juggler implements Performer {
    private int beanBags = 3;
    public Juggler() {}
    public Juggler(int beanBags) {
        this.beanBags = beanBags;
    }
    public void perform() throws PerformanceException {
        System.out.println("JUGGLING " + beanBags + " BEANBAGS");
    }
}
```

Chúng ta sẽ khai báo Duke trong file cấu hình Spring idol.xml :

```
<bean id="duke" class="Juggler" />
```

Về bản chất, duke được tạo ra bằng việc sử dụng đoạn mã lệnh Java sau đây :new Juggler();

Và sau đây là phần biểu diễn của duke :

```
ApplicationContext ctx = new ClassPathXmlApplicationContext(" idol.xml");
Performer performer = (Performer) ctx.getBean("duke");
```



```
performer.perform();
```

Kết quả in ra : JUGGLING 3 BEANBAGS

### 3.2. DI thông qua phương thức khởi tạo :

Cũng với lớp Juggler trên, ta hãy khai báo duke thông qua việc sử dụng constructor :

```
<bean id="duke" class="Juggler">
    <constructor-arg value="15" />
</bean>
```

Phần tử <constructor-arg> được sử dụng để cung cấp thêm thông tin cho Spring khi xây dựng bean. Nếu không có phần tử <constructor-arg> nào được khai báo thì phương thức khởi tạo mặc định sẽ được sử dụng.

Bây giờ, hãy xem Duke trình diễn, kết quả được in ra : JUGGLING 15 BEANBAGS

Trường hợp ta muốn tham chiếu đến đối tượng thông qua phương thức khởi tạo như ví dụ bên dưới.

```
public class PoeticJuggler extends Juggler {
    private Poem poem;
    public PoeticJuggler(Poem poem) {
        super ();
        this.poem = poem;
    }
    public PoeticJuggler(int beanBags, Poem poem) {
        super(beanBags);
        this.poem = poem;
    }
    public void perform() throws PerformanceException {
        super.perform();
        poem.recite();
    }
}

public interface Poem {
    void recite();
}

public class Sonnet29 implements Poem {
    private static String[] LINES = {"A","B","C"};
    public Sonnet29() {}
    public void recite() {
        for (int i = 0; i < LINES.length; i++) {
            System.out.println(LINES[i]);
        }
    }
}

<bean id="sonnet29" class="Sonnet29" />
<bean id="duke" class="PoeticJuggler">
    <constructor-arg value="15" />
    <constructor-arg ref="sonnet29" />
</bean>
```

Trong phần tử <constructor-arg > thứ hai ta không thể truyền vào một giá trị như bên trên được bởi vì đó không phải là kiểu dữ liệu đơn giản. Thay vào đó, ta sử dụng ref để chỉ ra rằng giá trị được truyền vào sẽ tham chiếu đến một bean khác thông qua ID của bean đó.

Hãy xem mã lệnh Java tương ứng :

```
Poem sonnet29 = new Sonnet29();
Performer duke = new PoeticJuggler(15, sonnet29);
```

**3.3.DI thông qua phương thức setter :**

```

public class Instrumentalist implements Performer {
    public Instrumentalist() {}
    public void perform() throws PerformanceException {
        System.out.print("Playing " + song + " : ");
        instrument.play();
    }
    private String song;
    public void setSong(String song) {
        this.song = song;
    }
    private Instrument instrument;
    public void setInstrument(Instrument instrument) {
        this.instrument = instrument;
    }
}

public interface Instrument {
    void play();
}

public class Saxophone implements Instrument {
    public Saxophone() {}
    public void play() {
        System.out.println("TOOT TOOT TOOT");
    }
}

<bean id="saxophone" class="Saxophone" />
<bean id="kenny" class="Instrumentalist">
    <property name="song" value="Jingle Bells" />
    <property name="instrument" ref="saxophone" />
</bean>

```

Như vậy, với cách khai báo trên, ngoài kenny có thể sử dụng bean saxophone thì các thành viên khác cũng có thể sử dụng nó khi cần.

**3.4.Inner beans :**

Trong ngôn ngữ lập trình Java, có lẽ chúng ta đã quá quen thuộc với khái niệm inner class. Spring cũng cung cấp kỹ thuật tương tự nhưng với tên gọi khác là inner bean, một loại bean được định nghĩa bên trong một bean khác.

```

<bean id="kenny" class="Instrumentalist">
    <property name="song" value="Jingle Bells" />
    <property name="instrument">
        <bean class="Saxophone" />
    </property>
</bean>

```

Cách khai báo này cho ta kết quả hoàn toàn giống với khai báo ở trên. Nhưng chỉ có duy nhất kenny mới có thể sử dụng bean Saxophone này mà thôi. Chú ý rằng, một inner bean không có thuộc tính ID. Inner bean không cung cấp khả năng sử dụng và chia sẻ mã lệnh, chỉ được sử dụng cho duy nhất một bean khác. Đó cũng là nhược điểm của loại bean này.

**3.5.DI trong Java Collections :**

Có 4 kiểu Java Collection có thể khai báo trong file cấu hình Spring thông qua XML bao gồm <list>, <set>, <map>, <props>.

```

public class OneManBand implements Performer {
    public OneManBand() {}
    public void perform() throws PerformanceException {

```

```

        for(Instrument instrument : instruments) {
            instrument.play();
        }
    }
    private Collection<Instrument> instruments;
    public void setInstruments(Collection<Instrument> instruments) {
        this.instruments = instruments;
    }
}

```

•Lists :

```

<bean id="hank" class="OneManBand">
    <property name="instruments">
        <list>
            <ref bean="guitar" />
            <ref bean="cymbal" />
            <ref bean="harmonica" />
        </list>
    </property>
</bean>

```

Phần tử <list> có thể bao gồm một hay nhiều giá trị và các giá trị này có thể trùng nhau. Ở đây phần tử <ref> được sử dụng để tham chiếu đến một bean khác. Phần tử <list> có thể sử dụng cho bất kỳ thuộc tính nào là thực thi của java.util.Collection hay thậm chí là một mảng. Hay nói cách khác, phần tử <list> vẫn làm việc khi thuộc tính instruments được khai báo như

```
java.util.List<Instrument> instruments;
```

Thậm chí là Instrument[] instruments;

•Sets :

```

<bean id="hank" class="OneManBand">
    <property name="instruments">
        <set>
            <ref bean="guitar" />
            <ref bean="cymbal" />
            <ref bean="harmonica" />
        </set>
    </property>
</bean>

```

Bất kỳ nơi nào sử dụng <list> thì cũng có thể sử dụng <set>. Nhưng <set> có một hiệu ứng lề hữu ích mà <list> không có đó là các giá trị trong <set> phải là duy nhất. Để thể hiện điều đó, dưới đây là một cách khai báo mới sử dụng <set> thay thế cho <list> :

```

<bean id="hank" class="OneManBand">
    <property name="instruments">
        <set>
            <ref bean="guitar" />
            <ref bean="cymbal" />
            <ref bean="harmonica" />
            <ref bean="harmonica" />
        </set>
    </property>
</bean>

```

Trong khai báo trên, thuộc tính instruments tham chiếu đến 2 bean harmonica, nhưng do chúng ta sử dụng <set> nên một bean harmonica sẽ bị bỏ qua.

•Maps :

```
public class OneManBand implements Performer {
    public OneManBand() {}
    public void perform() throws PerformanceException {
        for (String key : instruments.keySet()) {
            System.out.print(key + " : ");
            Instrument instrument = instruments.get(key);
            instrument.play();
        }
    }
    private Map<String,Instrument> instruments;
    public void setInstruments(Map<String,Instrument> instruments) {
        this.instruments = instruments;
    }
}

<bean id="hank" class="OneManBand">
    <property name="instruments">
        <map>
            <entry key="GUITAR" value-ref="guitar" />
            <entry key="CYMBAL" value-ref="cymbal" />
            <entry key="HARMONICA" value-ref="harmonica" />
        </map>
    </property>
</bean>
```

Mỗi phần tử <map> khai báo là một giá trị kiểu java.util.Map. Mỗi phần tử <entry> là một thành viên của <map>. Thuộc tính key là key của entry, và value-ref là thuộc tính tham chiếu đến một bean khác trong container.

Mặc dù trong ví dụ trên, ta sử dụng thuộc tính key để chỉ định một String và value-ref để chỉ định một giá trị tham chiếu, phần tử <entry> thực tế còn có hai thuộc tính để chỉ định cho key và value của entry nữa. Tất cả các thuộc tính đó được liệt kê trong bảng dưới đây.

Thuộc tính	Mục đích
Key	Chỉ định key như là một String
Key-ref	Chỉ định key như là một tham chiếu đến bean khác
Value	Chỉ định value như là một String
Value-ref	Chỉ định value như là một tham chiếu đến các bean khác

•Properties :

Cũng giống như <map>, phần tử <props> cũng bao gồm các cặp key-value, nhưng cả key và value đều là String.

```
public class OneManBand implements Performer {
    public OneManBand() {}
    public void perform() throws PerformanceException {
        for (Iterator iter = instruments.keySet().iterator();
            iter.hasNext();) {
            String key = (String) iter.next();
            System.out.println(key + " : " +
                instruments.getProperty(key));
        }
    }
    private Properties instruments;
    public void setInstruments(Properties instruments) {
        this.instruments = instruments;
    }
}
```

```

}
<bean id="hank" class="OneManBand">
  <property name="instruments">
    <props>
      <prop key="GUITAR">STRUM STRUM STRUM</prop>
      <prop key="CYMBAL">CRASH CRASH CRASH</prop>
      <prop key="HARMONICA">HUM HUM HUM</prop>
    </props>
  </property>
</bean>

```

### 3.6.Autowiring – Tự động móc nối :

Ở những phần trên, chúng ta đã tìm hiểu làm thế nào để tạo ra mối liên hệ giữa các bean thông qua các thao tác DI. Spring còn cung cấp kỹ thuật cho phép tự động liên kết móc nối các bean lại. Đó chính là Autowiring.

Spring cung cấp 4 loại Autowiring như sau :

◊byName : Tìm bean có id giống với tên thuộc tính để thực hiện DI

◊byType : Tìm những bean có kiểu giống với tên thuộc tính để thực hiện DI

◊constructor : Tìm những bean có kiểu giống với tên thuộc tính để thực hiện DI

◊autodetect : Autowire bởi constructor trước, sau đó là byName.

Autowiring byName :

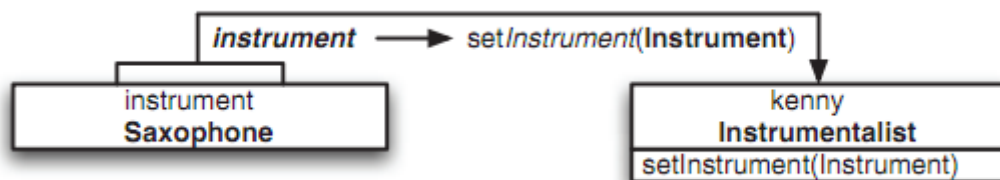
Trong Spring mọi thứ đều được đưa ra một cái tên. Các thuộc tính của bean cũng được xác định bởi tên và những bean được nối đến các thuộc tính đó cũng có một cái tên. Nếu tên của thuộc tính khớp với tên của bean được nối đến nó, điều đó gợi ý cho Spring rằng bean đó sẽ tự động nối đến thuộc tính đó.

```

<bean id="kenny" class="Instrumentalist" autowire="byName">
  <property name="song" value="Jingle Bells" />
</bean>

```

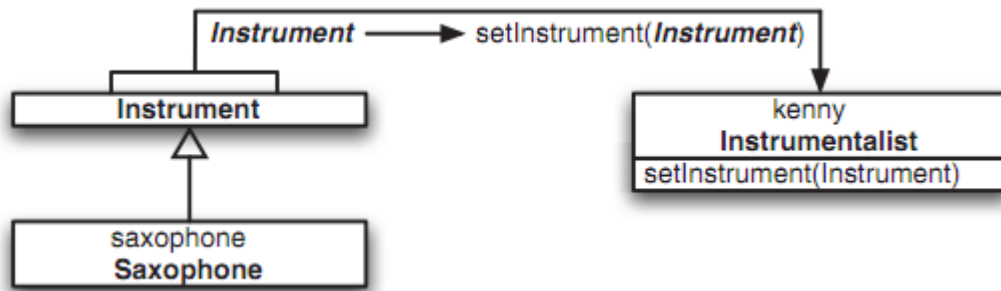
Thuộc tính “song” của kenny có giá trị là “Jingle Bells”. Ngoài thuộc tính này, kenny còn có thuộc tính có tên là “instrument” chưa được chỉ định một cách tường minh. Nhưng do chúng ta đã khai báo autowire là “byName” nên Spring container sẽ tìm ra bean có id là “instrument” để nối vào thuộc tính instrument. Nhược điểm của phương pháp này là nếu chúng ta khai báo nhiều bean kiểu Instrumentalist và sử dụng autowire byName thì tất cả sẽ có thuộc tính instrument tham chiếu đến cùng một bean có id là instrument.



Hình 3.4 - Autowiring byName

Autowiring byType :

Phương pháp này làm việc giống như phương pháp trên, ngoại trừ thay vì xem xét tên của thuộc tính thì kiểu của thuộc tính sẽ được xem xét. Spring sẽ tìm kiếm những bean có kiểu giống với kiểu của thuộc tính để thực hiện móc nối. Điều gì sẽ xảy ra nếu Spring tìm được nhiều hơn một bean có kiểu khớp với kiểu của thuộc tính. Khi đó Spring sẽ không thể quyết định được sẽ nên chọn bean nào và thay vào đó là một ngoại lệ sẽ được ném ra. Bởi vậy, phương pháp này tỏ ra không hữu hiệu trong ứng dụng thực tế.



Hình 3.5 - Autowiring byType

Autowiring constructor :

```
<bean id="duke" class="PoeticJuggler" autowire="constructor" />
```

Dựa vào khai báo trên, Spring sẽ tìm bean có kiểu khớp với một trong những tham số xuất hiện trong constructor của lớp PoeticJuggler.



Hình 3.6 - Autowiring constructor

Phương pháp này cũng có nhược điểm như autowiring byType. Hơn thế nữa, nếu có nhiều constructor có tham số khớp với các bean thì càng trở nên rắc rối hơn.

Autowiring autodetect :

Khai chúng ta muốn thực hiện mối nối giữa các bean nhưng không quyết định được nên sử dụng phương pháp nào thì bằng việc khai báo thuộc tính autowire là autodetect để Spring container quyết định thay cho ta.

```
<bean id="duke" class="PoeticJuggler" autowire="autodetect" />
```

### 3.7.Sự kế thừa giữa các bean :

Cũng như các lớp Java, giữa các bean trong Spring cũng có sự kế thừa. Trong đó một bean đóng vai trò là bean cha, bean còn lại đóng vai trò là bean con.

```
<bean id="baseSaxophonist" class="Instrumentalist" abstract="true">
```

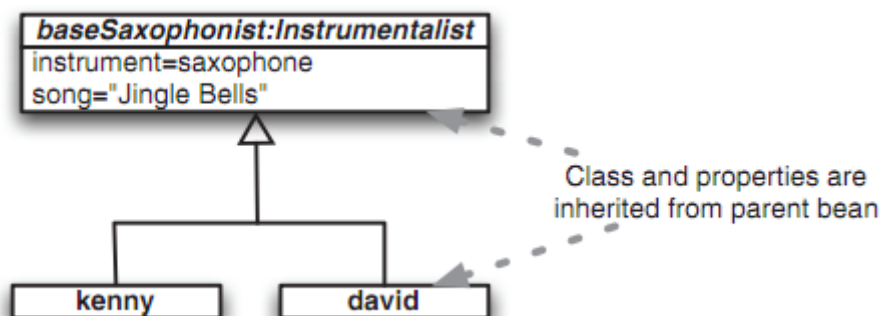
```
<property name="instrument" ref="saxophone" />
```

```
<property name="song" value="Jingle Bells" />
```

```
</bean>
```

```
<bean id="kenny" parent="baseSaxophonist" />
```

```
<bean id="david" parent="baseSaxophonist" />
```

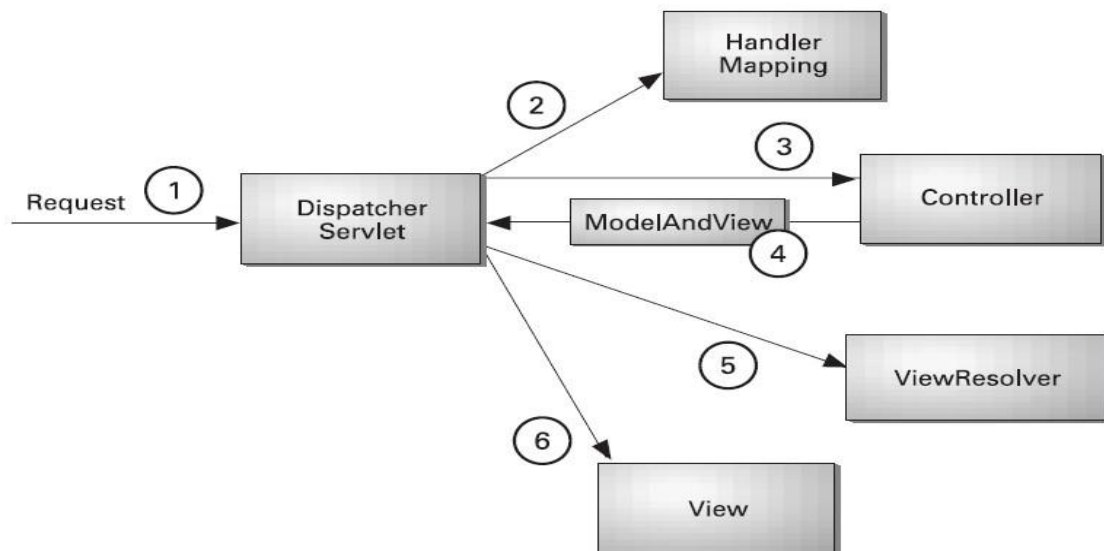


Hình 3.7 - Sự kế thừa giữa các Spring bean

Bean cha có thuộc tính abstract là true và bean con sử dụng thuộc tính parent để kế thừa từ bean cha.

## IV.Spring MVC :

### 4.1.Spring Workflow :



Hình 3.8 – Spring workflow

Mỗi khi user click vào link hoặc submit một form trên trình duyệt web. Yêu cầu (request) của user sẽ được gửi đi. Khi request này rời khỏi trình duyệt, mang theo các thông tin mà user yêu cầu. Trạm dừng đầu tiên của request là Dispatcher Servlet. Nhiệm vụ của Dispatcher Servlet là gửi request đến các Controller để xử lý. Nhưng thông thường trong một ứng dụng có thể có nhiều Controller. Dispatcher Servlet cần phải có sự trợ giúp để quyết định lựa chọn gửi đến Controller nào. Handler Mapping sẽ giúp Dispatcher Servlet đưa ra chặng dừng tiếp theo của request. Khi đã tìm ra Controller thích hợp với request. Dispatcher Servlet tiến hành gửi request đến Controller được lựa chọn. Tại Controller, request đợi Controller xử lý thông tin yêu cầu của user. Thông thường, Controller không trực tiếp làm công việc này mà ủy thác cho một đối tượng chuyên dụng khác, ta gọi các đối tượng này là các đối tượng dịch vụ (service object). Dữ liệu thu được sau khi xử lý request được mang lại cho user và hiển thị lên trình duyệt, dữ liệu này được quy vào Model. Để hiển thị đến được user, Controller đóng gói dữ liệu và tên của một View vào trong đối tượng ModelAndView. Sau đó gửi lại cho Dispatcher Servlet. Như vậy, trong đối tượng ModelAndView này bao gồm dữ liệu và một View nhưng View này chỉ là gợi ý. Đối tượng ModelAndView không hề mang tên của View thật sự mà nó chỉ mang một cái tên logic, dựa vào cái tên này sẽ tìm ra được View thực sự. Một lần nữa, Dispatcher Servlet phải hỏi View Resolver để giúp tìm ra View thực sự đang cần tìm. Đến đây, Dispatcher Servlet đã biết chính xác View nào sẽ được chọn, việc làm cuối cùng không có gì phức tạp, đó là hiển thị kết quả cho user. Như vậy, chặng dừng cuối cùng của request là nơi mà View được thực hiện, thông thường đó là trang JSP.

### 4.2.Cấu hình DispatcherServlet :

Trái tim của Spring MVC chính là DispatcherServlet. Vì vậy, trước khi có thể sử dụng Spring MVC, chúng ta phải cấu hình cho DispatcherServlet. Giống như mọi Servlet khác, DispatcherServlet cũng được cấu hình trong web.xml.

```

<servlet>
  <servlet-name>dispatcher</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>

```

Trong đó <servlet-name> là tên của servlet. Mặc định, khi DispatcherServlet được load thì nó cũng sẽ load Spring application context từ một file XML có tên dựa trên tên của nó. Trong trường hợp này, do servlet có tên là “dispatcher” nên DispatcherServlet sẽ load file dispatcher-servlet.xml. Chúng ta sẽ tìm hiểu file XML này trong phần sau. Công việc tiếp theo, chúng ta phải chỉ ra địa chỉ URL nào sẽ được điều khiển bởi DispatcherServlet.

```

<servlet-mapping>
  <servlet-name>dispatcher</servlet-name>
  <url-pattern>*.htm</url-pattern>
</servlet-mapping>

```

Chúng ta chọn \*.htm vì đây là pattern chuẩn cho hầu hết các ứng dụng Spring MVC.

Đến đây, DispatcherServlet đã được cấu hình xong và đã đưa ra ánh xạ URL. Chúng ta đã có thể bắt đầu viết các chức năng cho ứng dụng. Nhưng vẫn còn một việc dưới đây được khuyến khích nên thêm vào web.xml, đó là cấu hình cho Application context loader.

#### 4.3. Cấu hình Application Context :

Như đã đề cập ở phần trên, DispatcherServlet sẽ load Spring application Context từ một file XML dựa vào servlet-name. Chúng ta có thể để tất cả các bean của ứng dụng trong file XML này. Tuy nhiên điều có không có nghĩa là chúng ta không thể phân chia Spring application Context thành nhiều file XML khác nhau. Mà thông thường, một ứng dụng Spring thường có nhiều file XML để cấu hình cho ứng dụng. Nhưng tất cả các file này chỉ nên được đặt tập trung ở một layer duy nhất của ứng dụng mà thôi. Bởi vì nó sẽ làm cho ứng dụng dễ dàng quản lý và đặc biệt là dễ bảo trì. Ngoài ra khi hoán đổi cấu hình tại một layer thì các layer khác sẽ không bị ảnh hưởng (giả sử ta thay đổi cấu hình của một lớp bằng Hibernate sang sử dụng iBATIS cũng là một ORM framework chẳng hạn ).

Để chắc chắn rằng tất cả các file cấu hình cho ứng dụng được load, chúng ta cần cấu hình context loader trong file web.xml. Một context loader sẽ đảm nhận trách nhiệm load những file cấu hình với điều kiện khi DispatcherServlet được load. Thường sử dụng nhất là servlet listener có tên ContextLoaderListener được cấu hình trong web.xml như dưới đây :

```

<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>

```

Khi đã cấu hình ContextLoaderListener , chúng ta cần chỉ ra vị trí các file cấu hình sẽ được load cùng DispatcherServlet . Nếu việc này không được chỉ định, Spring context loader sẽ tìm kiếm các file cấu hình Spring tại /WEB-INF/applicationContext.xml.

```

<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/applicationContext.xml</param-value>
</context-param>

```

Tuy nhiên ta hoàn toàn có thể chỉ định một danh sách các đường dẫn đến các file cấu hình Spring .

```

<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    /WEB-INF/dispatcher-service.xml,
    /WEB-INF/dispatcher-data.xml,
    /WEB-INF/dispatcher-security.xml
  </param-value>
</context-param>

```

Đến đây DispatcherServlet đã được cấu hình xong và đã sẵn sàng có thể gửi các yêu cầu đến web layer của ứng dụng. Nhưng web layer hiện tại chưa được xây dựng. Việc tiếp theo mà chúng ta phải thực hiện là xây dựng nó.

#### 4.4. Xây dựng web layer :

Để dễ dàng hình dung, chúng ta đi vào ví dụ thực tế sau đây.

Mỗi ứng dụng web đều có một hay nhiều form. Sau đây là 4 bước cơ bản ngắn nhất để tạo ra một form trong Spring MVC.

1. Viết 1 lớp controller để đảm nhiệm các chức năng logic của form.
2. Cấu hình controller này trong cấu hình context của DispatcherServlet, chính là file dispatcher-servlet.xml.

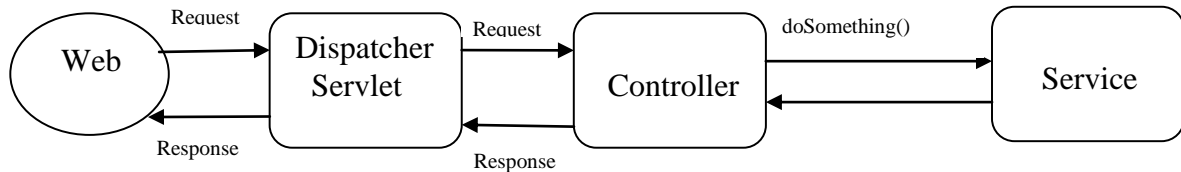


3.Cấu hình View Resolver để liên kết controller đến trang JSP.

4.Viết trang JSP để hiển thị form cho user.

◊Thiết kế Controller :

Controller là một class bảo đảm sự giao tiếp của người sử dụng với các chức năng của ứng dụng. Hay nói cách khác đó là nơi tiếp nhận và xử lý request từ người sử dụng.



Hình 3.9 - Mô hình xử lý của một controller

Controller thường không trực tiếp xử lý yêu cầu của người sử dụng mà nó ủy thác cho một đối tượng khác thực hiện , ta gọi những đối tượng đó là các service object. Và cuối cùng, Controller lấy kết quả nhận được gửi về cho trình duyệt web hiển thị lên cho người sử dụng.

Ví dụ sau đây là một Controller cho trang tìm kiếm thông tin về các dự án trong chương trình quản lý các dự án của một công ty phần mềm.

```

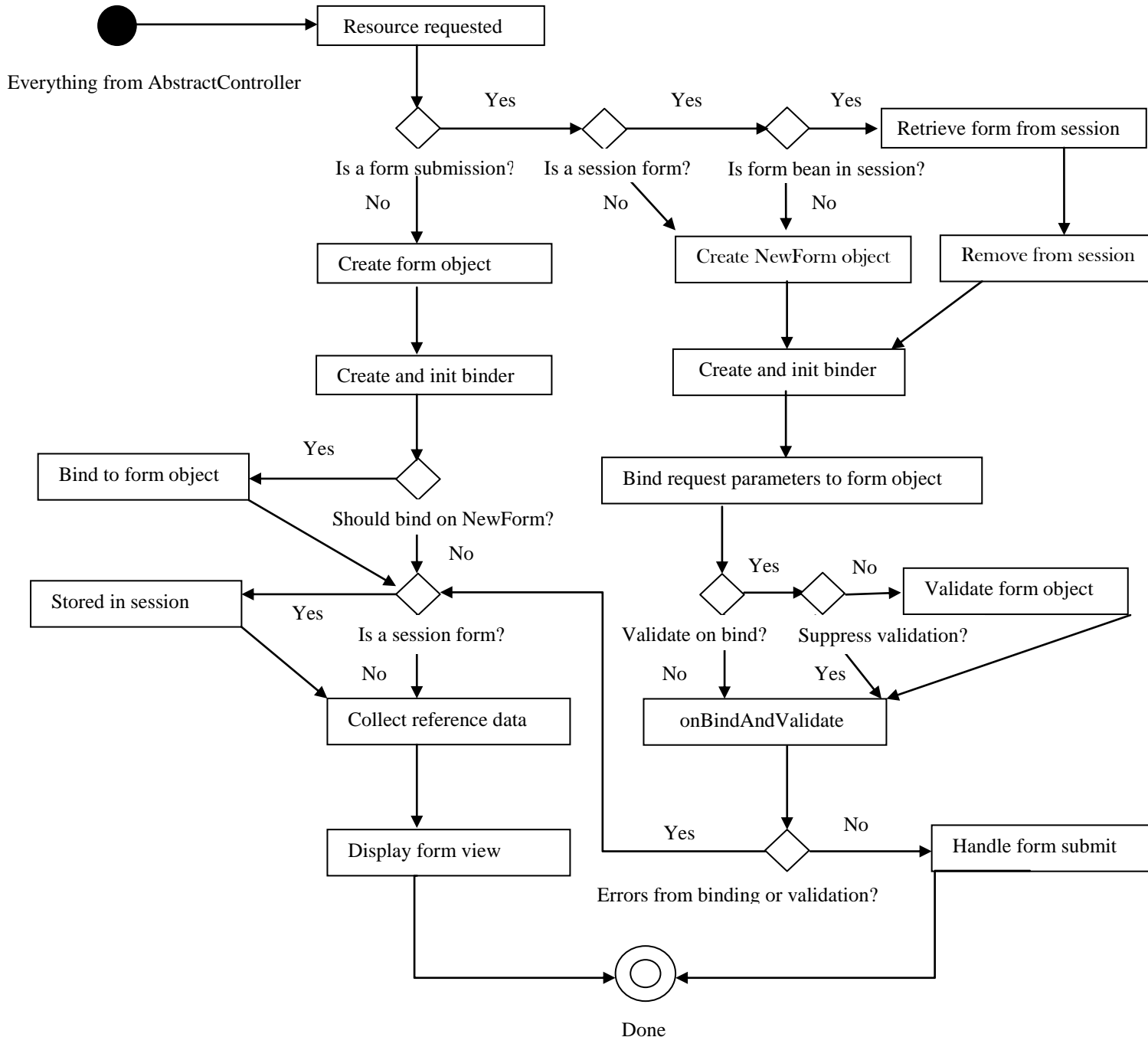
public class SearchProjectsController extends SimpleFormController {
    private ProjectService projectService;
    public SearchProjectsController() {
        super();
        setSessionForm(true);
        setCommandName("query");
        setCommandClass(ProjectQuery.class);
        setFormView(UrlConstants.SEARCH_PROJECTS_URL);
        setSuccessView(UrlConstants.SEARCH_PROJECTS_URL);
    }
    @Override
    protected ModelAndView onSubmit(HttpServletRequest request,
        HttpServletResponse response, Object command, BindException errors)
        throws Exception {
        ProjectQuery query = (ProjectQuery) command;
        List<Project> projects = getProjectService().findByQuery(query);
        if (projects.isEmpty()) {
            errors.reject("search.project.empty");
        } else {
            request.setAttribute("projects", projects);
        }
        return showForm(request, response, errors);
    }
    .....
}
  
```

Controller trên được kế thừa từ lớp SimpleFormController, là một trong những controller phổ biến trong Spring MVC. Đối tượng projectService là đối tượng được controller ủy thác thực thi nhiệm vụ của mình.

Chúng ta sẽ tìm hiểu một số phương thức đặc biệt trong controller này .

- setSessionForm() : Chỉ ra rằng form có được lưu vào trong session khi user yêu cầu một form mới hay không.
- setCommandName () : Thông thường, trong một Spring form luôn có một đối tượng đặc biệt gọi là command object , đối tượng này là duy nhất và sử dụng để tiếp nhận các yêu cầu từ phía user.
- setCommandClass() : Chỉ định kiểu của đối tượng command.

- setFormView() : Đưa ra một cái tên logic của một view mặc định của controller. Dựa vào tên này, View Resolver sẽ xác định được chính xác view được chọn để hiển thị kết quả.
  - setSuccessView () : Chỉ ra view nào sẽ được chọn sau khi quá trình user submit thành công.
  - onSubmit () : Phương thức xử lý sự kiện submit mà user tạo ra trên form.
  - showForm() : Sau khi các chức năng logic được hoàn thành bởi các service object, đó là thời gian controller gửi kết quả trở lại cho trình duyệt thông qua phương thức showForm(). Phương thức này sẽ trả về một đối tượng ModelAndView. Đối tượng này là một khái niệm quan trọng trong Spring. Nó sẽ đóng gói view và toàn bộ dữ liệu của view đó.
- Sau đây là sơ đồ hoạt động của SimpleFormController.



•initBinder() : Phương thức này sẽ được gọi ngay sau khi thực thể form được tạo ra và controller sử dụng để tạo ra sự gắn kết dữ liệu cho đối tượng trong form. Thông thường các lập trình viên sẽ sử dụng phương thức này để đăng ký cái được gọi chung là PropertyEditors của một form. Ví dụ như đăng ký định dạng cho các đối tượng kiểu ngày, thời gian, các đối tượng Java Collection hay các đối tượng làm nhiệm vụ upload file...

•referenceData() : Phương thức này có nhiệm vụ cung cấp dữ liệu cho các thuộc tính của đối tượng command.

◊Cấu hình bean controller :

Như vậy ta đã thiết kế controller xong. Kế tiếp là việc cấu hình bean controller trong file cấu hình của DispatcherServlet (dispatcher-servlet.xml ).

```
<bean id="searchProjectsController" class="SearchProjectsController">
    <property name="projectService" ref="projectService" />
</bean>
```

Chú ý rằng, không chỉ riêng bean projectService mà tất cả các bean của các service object khác đều phải nằm trong Application Context. Hay nói cách khác, các bean này sẽ được khai báo trong applicationContext.xml.

```
<bean id="projectService" class="ProjectService">
    <property name="projectDao" ref="projectDao" />
</bean>
```

Thực ra, cũng như controller, các service object cũng ủy thác nhiệm vụ được giao cho một đối tượng khác thực hiện đó là các DAO hay Data Access Object. Các đối tượng này có khả năng làm việc trực tiếp với cơ sở dữ liệu của ứng dụng. Và chúng cũng được khai báo trong applicationContext.xml.

```
<bean id="projectDao" class="ProjectDao">
    <property name="sessionFactory">
        <ref local="sessionFactory" />
    </property>
</bean>
```

Để có thể làm việc trực tiếp với cơ sở dữ liệu, các DAO được cung cấp một thuộc tính chuyên đảm nhận việc kết nối, thực hiện truy vấn đến cơ sở dữ liệu. Đó chính là thuộc tính có tên sessionFactory mà chúng ta đã tìm hiểu qua trong Hibernate framework.

Spring MVC được thiết kế để làm việc tốt với các framework khác, đặc biệt là hỗ trợ đầy đủ cho Hibernate. Spring MVC sẽ sử dụng Hibernate như là một cầu nối đến cơ sở dữ liệu. Tuy nhiên, cũng xin nói thêm, trong Spring MVC cũng hỗ trợ sẵn khả năng làm việc trực tiếp với cơ sở dữ liệu nhưng chắc chắn không thể mạnh bằng khi kết hợp với các ORM framework khác.

Sau đây là khai báo bean sessionFactory trong Spring Application Context có kết hợp với Hibernate.

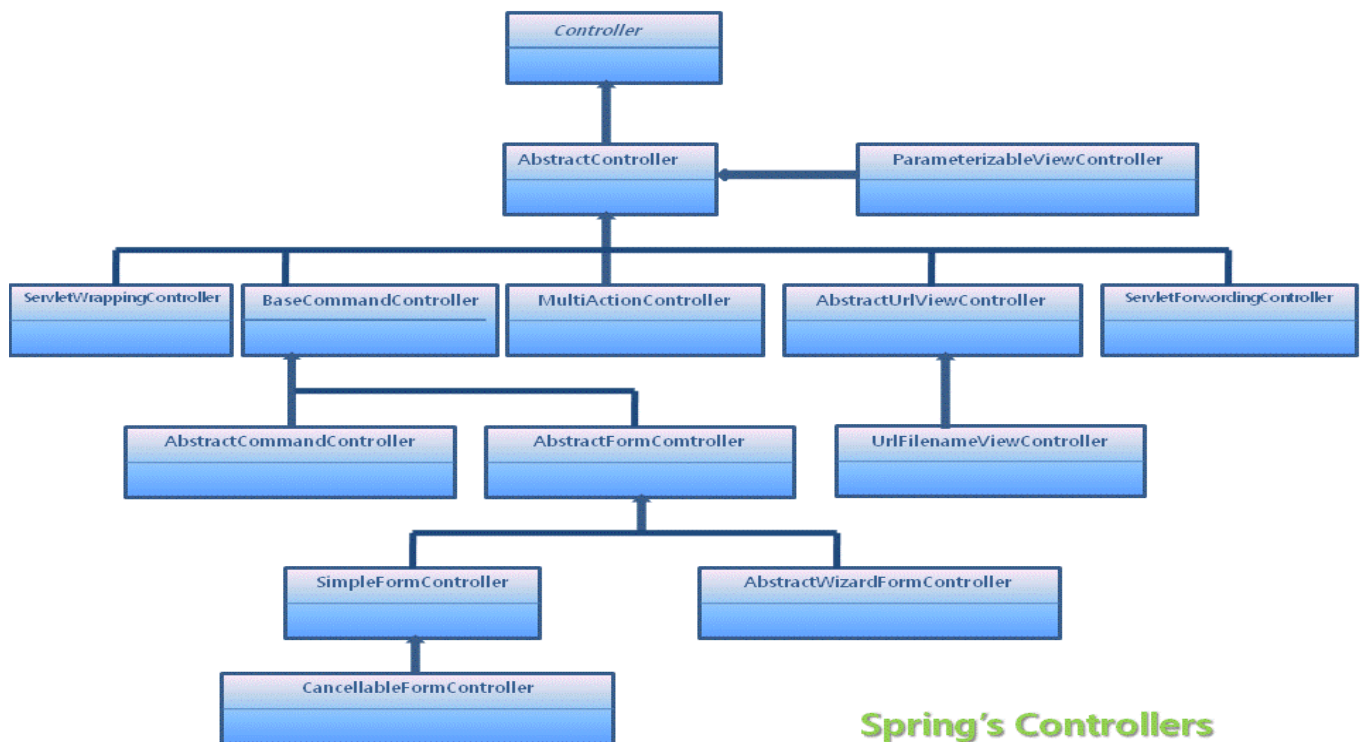
```
<bean id="abstractSessionFactory" abstract="true"
    class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
    <property name="configLocation">
        <value>/WEB-INF/hibernate.cfg.xml</value>
    </property>
    <property name="configurationClass">
        <value>org.hibernate.cfg.Configuration</value>
    </property>
</bean>
<bean id="sessionFactory" parent="abstractSessionFactory" />
```

Bean abstractSessionFactory là một đối tượng của lớp LocalSessionFactoryBean. Đây là một lớp được Spring thiết kế để có thể liên kết với Hibernate. Lớp này đóng vai trò tương tự như sessionFactory trong Hibernate. Thuộc tính abstract là true để chỉ định đây là bean trừu tượng và các bean khác có thể extends từ nó. Cách làm này sẽ giúp ta có thể tạo ra nhiều hơn

một SessionFactory trong cùng một ứng dụng. Thuộc tính configLocation chỉ định đường dẫn đến file hibernate.cfg.xml . Mọi kết nối sẽ được thiết lập thông qua nó. Trong trường hợp ta sử dụng phương pháp ánh xạ XML thì thuộc tính configurationClass có giá trị là org.hibernate.cfg.Configuration và org.hibernate.cfg.AnnotationConfiguration khi sử dụng Annotations Mapping.

Trong khai báo bean sessionFactory , thuộc tính parent có giá trị là abstractSessionFactory để chỉ ra rằng nó được kế thừa từ abstractSessionFactory.

Bên cạnh SimpleFormController, Spring MVC còn cung cấp rất nhiều controller khác nữa. Mỗi controller được thiết kế cho những nhiệm vụ đặc biệt. Sau đây là danh sách phân cấp của tất cả các controller trong Spring MVC.



Hình 3.11 - Các controller trong Spring MVC

◇Cấu hình View Resolver :

Khi kết quả được gửi đến trình duyệt, để chỉ định chính xác trang JSP nào sẽ hiển thị chúng, ta phải khai báo bean viewResolver trong dispatcher-servlet.xml. Nhiệm vụ của bean này rất đơn giản là lấy tên logic của view trong đối tượng ModelAndView và ánh xạ nó đến một view thực sự.

```

<bean id="viewResolver"
      class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass" value="org.springframework.web.servlet.view.JstlView" />
    <property name="prefix">
        <value>/WEB-INF/jsp/</value>
    </property>
    <property name="suffix">
        <value>.jsp</value>
    </property>
</bean>
  
```

Spring MVC cung cấp nhiều đối tượng làm nhiệm vụ View Resolver nhưng ta sẽ sử dụng lớp InternalResourceViewResolver , là View Resolver đơn giản nhất. Bean viewResolver sẽ

cho biết nơi lưu trữ các view và định dạng của các view đó. Thông thường, các view được lưu trữ tại thư mục /WEB-INF/jsp và định dạng của chúng là .jsp, một trong những loại view phổ biến trong Spring MVC.

Tiếp theo chúng ta khai báo ánh xạ URL, việc làm này cho Spring biết được tương ứng với mỗi controller thì có những view thích hợp nào dành cho chúng.

```
<bean id="urlMapping"
class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="interceptors">
        <list>
            <bean id="localeChangeInterceptor"
class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor" />
        </list>
    </property>
    <property name="mappings">
        <props>
            <prop key="searchProjects.htm">searchProjectsController</prop>
            .....
        </props>
    </property>
</bean>
```

Bean urlMapping này có nhiệm vụ ánh xạ từ url đến controller. Ở trên, tất cả các thao tác của user trên url “searchProjects.htm” sẽ được controller searchProjectsController xử lý. Thuộc tính interceptors là một tính năng được Spring MVC cung cấp nhằm giải quyết vấn đề đa ngôn ngữ trong ứng dụng. Hay nói một cách dễ hiểu, thuộc tính này sẽ giúp cho ứng dụng có thể chuyển đổi qua lại giữa nhiều ngôn ngữ khác nhau kể cả trên giao diện và dữ liệu chương trình.

Đến đây, chúng ta đã cấu hình toàn bộ cho Spring MVC xong. Việc làm tiếp theo rất đơn giản là thiết kế các trang JSP để hiển thị kết quả.

◊Thiết kế View :

Việc thiết kế trang JSP trong Spring cũng khác so với việc thiết kế các trang JSP bình thường. JSP trong Spring được hỗ trợ thêm các thư viện thẻ gọi là các taglib. Sau đây là các thư viện thẻ thường gặp.

□Thư viện : <%@ taglib uri="http://www.springframework.org/tags" prefix="spring"%>

- Thẻ <spring:message> : Xuất ra một thông điệp nằm trong file \*.properties.

□Thư viện : <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>

- Thẻ <form:form> : Khai báo 1 Spring form
- Thẻ <form:input> : Khai báo 1 Spring input
- Thẻ <form:select> : Khai báo 1 Spring select list
- Thẻ <form:checkbox> : Khai báo 1 Spring checkbox

□Thư viện : <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

- Thẻ <c:choose> : Biểu thức switch
- Thẻ <c:when> : Biểu thức case
- Thẻ <c:otherwise> : Biểu thức default trong mệnh đề switch case
- Thẻ <c:if> : Biểu thức điều kiện if
- Thẻ <c:out> : Xuất ra giá trị 1 biến là một attribute trong request
- Thẻ <c:forEach> : Thực hiện 1 vòng lặp

□Thư viện : <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>

- Thẻ <fmt:message> : Xuất ra một thông điệp nằm trong file \*.properties.
- Thẻ <fmt:formatDate> : Định dạng cách hiển thị dữ liệu ngày, thời gian.
- Thẻ <fmt:formatNumber> : Định dạng cách hiển thị dữ liệu số.

Chúng ta tiếp tục với chặng đường thiết kế một Spring form . Trang searchProjects.jsp :

```

<form:form action="" method="post" commandName="query">
<form:input path="searchKey" />
<input type="submit" value='<spring:message code="search"/>'/>
<c:choose>
  <c:when test="${countList == 0 && searchKey != ""}>
    <spring:message code="search.empty"/>
  </c:when>
  <c:otherwise>
    <c:forEach var="project" items="${projects}">
      <form:checkbox path="projectIds" value="${project.projectid}" />
      <a href="project.htm?pnum=${project.number}">
        <c:set value="${fn:trim(project.logourl)}" var="logo"/>
        <c:choose>
          <c:when test="${logo == null || logo == ""}>
            
          </c:when>
          <c:otherwise></c:otherwise>
        </c:choose>
      </a>
    <table>
      <tr>
        <td>
          <spring:message code="project.number"/>
          <c:out value="${project.number}"></c:out>
        </td>
        <td>
          <spring:message code="project.customer"/>
          <c:out value="${project.customer.name}"></c:out>
        </td>
      </tr>
      <tr>
        <td>
          <spring:message code="project.group"/>
          <c:out value="${project.group.employee.visa}"></c:out>
        </td>
        <td>
          <spring:message code="project.status"/>
          <c:out value="${project.projectStatus.name}"></c:out>
        </td>
      </tr>
      <tr>
        <td>
          <spring:message code="project.startdate"/>
          <c:out value="${project.startdate}"></c:out>
        </td>
      </tr>
    </table>
    </c:forEach>
  </c:otherwise>
</c:choose>
</form:form>

```

Trong thẻ <form:form>, thuộc tính commandName chính là tên command object của controller. Đối tượng này là một thực thể của lớp ProjectQuery do ta định nghĩa như sau.

```
public class ProjectQuery {
    private String searchKey;
    private List<Long> projectIds;
}
```

Lớp này có hai thuộc tính, searchKey là từ khóa tìm kiếm, từ khóa này được gán vào thẻ <form:input> thông qua path=”searchKey”. Thuộc tính projectIds là danh sách lưu trữ các giá trị id của các project trong kết quả của việc tìm kiếm và danh sách này được gán vào thẻ <form:checkbox> thông qua việc khai báo path=”projectIds”. Khi user nhập từ khóa vào textbox hoặc click chọn vào các checkbox thì các giá trị này sẽ tự động gán cho các thuộc tính của command object. Kỹ thuật này giúp chúng ta tiết kiệm được việc phải viết mã lệnh rất nhiều.

Để sử dụng thẻ <spring:message>, ta phải tạo ra file properties chứa giá trị của các thuộc tính code cho thẻ này. Mặc định thì Spring sẽ tìm các code này trong file có tên messages.properties. Và sau đây là nội dung của nó.

```
search=Tìm kiếm
search.empty=Không tìm thấy kết quả nào
project.number=Mã Số
project.customer=Khách Hàng
project.group=Nhóm
project.status=Trạng Thái
project.startdate=Ngày Bắt Đầu
project.endDate=Ngày Kết Thúc
.....
```

Và sau đây là kết quả thu được :

Danh Sách Các Dự Án
+ Thêm Dự Án | Xóa Dự Án

<input type="checkbox"/>			<a href="#">Sửa</a> <b>ProGate</b>	<div style="display: flex; justify-content: space-between;"> <div> <p>Mã Số : 345</p> <p>Nhóm : vydx</p> <p>Ngày Bắt Đầu : 2011-08-31</p> </div> <div> <p>Khách Hàng : HP</p> <p>Trạng Thái : Invalidate</p> </div> </div>
<input type="checkbox"/>			<a href="#">Sửa</a> <b>ProSim</b>	<div style="display: flex; justify-content: space-between;"> <div> <p>Mã Số : 34</p> <p>Nhóm : vydx</p> <p>Ngày Bắt Đầu : 2011-07-23</p> </div> <div> <p>Khách Hàng : IBM</p> <p>Trạng Thái : Invalidate</p> </div> </div>

Hiện Thị 1 - 2 / 2 Kết Quả

Số Item Mỗi Trang

Trang

⏪ Đầu
⏴ Trước
Sau
⏵ Cuối

Hình 3.12 - Form tìm kiếm dự án

Như vậy, chúng ta đã tìm hiểu về những điều cơ bản nhất về cả hai framework Hibernate và Spring MVC. Tuy nhiên còn rất nhiều điều khác nữa để nói về chúng. Nhưng giới hạn không cho phép nên chúng ta sẽ nghiên cứu vào những bài báo cáo sau. Trong phần tiếp theo sẽ là một hình cơ sở dữ liệu của chương trình demo “Chương trình quản lý thông tin các dự án phần mềm”.



**CHƯƠNG IV****MÔ HÌNH CƠ SỞ DỮ LIỆU****I.Công cụ thiết kế :**

- Hệ quản trị cơ sở dữ liệu MySQL server 5.1
- MySQL workbench 5.2
- Power Designer 12.5

**II.Danh sách các bảng :**

Qui ước :

PK : Khóa chính

FK : Khóa ngoại

M : Thành phần bắt buộc phải có

O : Tùy chọn

(\*) : Thông tin thêm

**1.Bảng company : Chứa các thông tin chung về công ty**

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	companyid	Bigint(20)	PK	M
2	shortname	Varchar(10)	Ký hiệu	M
3	name	Varchar(100)	Tên đầy đủ	M
4	establisheddate	Date	Ngày thành lập	O
5	logourl	Varchar(200)	Logo	O
6	slogan	Varchar(200)	Slogan	O
7	videourl	Varchar(200)	Video giới thiệu	O
8	address	Varchar(100)	Địa chỉ	O
9	email	Varchar(50)	Email	M
10	phone	Varchar(20)	Số điện thoại	O
11	fax	Varchar(20)	Số fax	O
12	achievement	Text	Thành tựu	O
13	overview	Text	Tổng quan	O
14	career	Text	Nghề nghiệp	O
15	technology	Text	Công nghệ	O

(\*) Chương trình sẽ có 1 trang chủ, trên trang này ngoài việc hiển thị các thông tin về công ty còn có 1 form đăng nhập. Khi nhân viên đăng nhập thành công sẽ dẫn vào trang menu các ứng dụng. Các ứng dụng sẽ được lưu trữ trong bảng application bên dưới đây.

**2.Bảng application : Chứa thông tin về các ứng dụng**

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	applicationid	Bigint(20)	PK	M
2	title	Varchar(50)	Tiêu đề	M
3	imageurl	Varchar(200)	Ảnh nền	M
4	iconurl	Varchar(200)	Icon	O
5	friendlyurl	Varchar(100)	URL	M

(\*) Ứng dụng chính của chương trình là quản lý thông tin các dự án, bên cạnh đó còn có quản lý thông tin nhân viên, thông tin khách hàng/đối tác.

**3.Bảng employee : Chứa thông tin về các nhân viên**

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	employeeid	Bigint(20)	PK	M
2	departmentid	Bigint(20)	FK – Phòng ban	M

3	rankid	Bigint(20)	FK – Thứ hạng	M
4	universityid	Bigint(20)	FK – Trường	M
5	marriedstatus	Varchar(20)	Tình trạng hôn nhân	O
6	diplomaid	Bigint(20)	FK – Bằng cấp	M
8	visa	Varchar(10)	Tên viết tắt	M
9	firstname	Varchar(10)	Tên nhân viên	M
10	lastname	Varchar(50)	Họ nhân viên	M
11	dateofbirth	Date	Ngày sinh	M
12	placeofbirth	Varchar(100)	Nơi sinh	O
13	hiredDate	Date	Ngày vào làm	O
14	phone	Varchar(20)	Số điện thoại	O
15	email	Varchar(50)	Email	M
16	yahoo	Varchar(50)	Yahoo nickname	O
17	bankaccountno	Varchar(20)	Số TK ngân hàng	O
18	gender	Boolean	Giới tính	O
19	salary	Float	Lương	O
20	imageurl	Varchar(200)	Ảnh nhân viên	O
21	available	Boolean	F : nghỉ việc	M

4. Bảng rank : Thứ hạng của nhân viên

VD: Engineer, Manager, Senior Manager

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	rankid	Bigint(20)	PK	M
2	name	Varchar(30)	Tên	M

5. Bảng university : Tên các trường đại học mà nhân viên đã tốt nghiệp

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	universityid	Bigint(20)	PK	M
2	name	Varchar(100)	Tên	M

6. Bảng diploma : Tên các loại văn bằng của nhân viên

VD : Cử nhân , Kỹ sư, Thạc sỹ...

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	Diplomaid	Bigint(20)	PK	M
2	Name	Varchar(30)	Tên	M

7. Bảng department : Danh sách các phòng ban trong công ty

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	department id	Bigint(20)	PK	M
2	name	Varchar(30)	Tên	M
3	email	Varchar(50)	Email	M
4	phone	Varchar(20)	Số điện thoại	O

8. Bảng role : Danh sách các quyền của người dùng

VD : Admin, Manager, HR, Developer...

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	roleid	Bigint(20)	PK	M
2	name	Varchar(30)	Tên	M

9.Bảng user : Thông tin người dùng

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	userid	Bigint(20)	PK	M
2	roleid	Bigint(20)	PK – Vai trò	M
3	employeeid	Bigint(20)	PK – Nhân viên	M
4	loginname	Varchar(30)	Tên đăng nhập	M
5	password	Varchar(50)	Mật khẩu	M
6	email	Varchar(50)	Email	M
7	enable	Boolean	Enable/Disable	M
8	imageurl	Varchar(200)	Ảnh đại diện	O

10.Bảng customer : Chứa các thông tin về khách hàng

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	customerid	Bigint(20)	PK	M
2	fieldid	Bigint(20)	FK – Lĩnh vực hoạt động	M
3	typeid	Bigint(20)	FK – Loại hình tổ chức	M
4	cityid	Bigint(20)	FK – Thành phố	M
5	name	Varchar(30)	Tên khách hàng	M
6	representator	Varchar(50)	Người đại diện	M
7	yahoo	Varchar(50)	Yahoo nickname	M
8	address	Varchar(100)	Địa chỉ	O
9	email	Varchar(50)	Email	M
10	phone	Varchar(20)	Số điện thoại	O
11	website	Varchar(50)	Website	O
12	logourl	Varchar(200)	Logo	O
13	slogan	Varchar(100)	Slogan	O
14	description	Text	Thông tin mô tả về khách hàng	O
15	isfriendly	Boolean	Đánh dấu là khách hàng thân thiết	O

11.Bảng customertype : Loại hình tổ chức của khách hàng

VD : Công ty TNHH, công ty vốn đầu tư nước ngoài, tổ chức chính phủ...

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	typeid	Bigint(20)	PK	M
2	name	Varchar(100)	Tên	M

12.Bảng customerfield : Lĩnh vực hoạt động của khách hàng

VD : Ngân hàng, bảo hiểm, giao thông vận tải, kinh doanh...

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	fieldid	Bigint(20)	PK	M
2	name	Varchar(100)	Tên	M

13.Bảng city : Chứa danh sách các thành phố

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	cityid	Bigint(20)	PK	M
2	name	Varchar(30)	Tên	M
3	countryid	Bigint(20)	PK	M

14.Bảng country : Chứa danh sách các quốc gia

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	countryid	Bigint(20)	PK	M
2	name	Varchar(30)	Tên	M
3	sign	Varchar(10)	Ký hiệu	M
4	code	Varchar(10)	Mã nước	M

15.Bảng project : Lưu trữ các dự án

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	projectid	Bigint(20)	PK	M
2	customerid	Bigint(20)	FK – Khách hàng	M
3	statusid	Bigint(20)	FK – Trạng thái	M
4	groupid	Bigint(20)	FK - Nhóm	M
5	number	Int(11)	Số dự án	M
6	name	Varchar(100)	Tên dự án	M
7	startdate	Date	Ngày bắt đầu	M
8	enddate	Datet	Ngày kết thúc	O
9	logourl	Varchar(200)	Logo	O

16.Bảng projectstatus : Chứa các trạng thái của dự án

VD : Invalidate , To Validate, Validated, Finished

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	statusid	Bigint(20)	PK	M
2	name	Varchar(100)	Tên	M

17.Bảng task : Chứa các công việc của các dự án

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	taskid	Bigint(20)	PK	M
2	categoryid	Bigint(20)	FK – Loại	M
3	projectid	Bigint(20)	FK – Dự án	M
4	statusid	Bigint(20)	FK – Trạng thái	M
5	priorityid	Bigint(20)	FK – Độ ưu tiên	M
6	severityid	Bigint(20)	FK – Độ phức tạp	M
7	reporter	Varchar(10)	Người tạo	M
8	assignedto	Varchar(10)	Người nhận	O
9	code	Varchar(20)	Mã công việc	M
10	summary	Varchar(10)	Tóm tắt	M
11	description	Text	Mô tả cụ thể	M
12	additionalinformation	Text	Thông tin thêm	O
13	percentcomplete	Int(11)	% hoàn thành	O
14	createdtime	Datetime	Thời gian tạo	O
15	starttime	Datetime	Thời gian bắt đầu	O
16	endtime	Datetime	Thời gian kết thúc	O
17	estimatetime	Float	Thời gian ước lượng	O
18	actualtime	Float	Thời gian thực tế	O
19	phase	Varchar(30)	Pha	O
20	attachedfiles	Text	Files đính kèm	O

21	lastchanged	Datetime	Cập nhật cuối	O
22	note	Text	Ghi chú	O

18.Bảng taskcategory : Loại công việc

VD : Bug, Change – Request, New-Feature, Customer Reported Bug, Deployment...

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	categoryid	Bigint(20)	PK	M
2	name	Varchar(30)	Tên	M

19.Bảng taskstatus: Trạng thái của công việc

VD : Pending, Assigned, Confirmed , Feedback ,To be reviewed, Resolved ,Closed

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	statusid	Bigint(20)	PK	M
2	name	Varchar(30)	Tên	M
3	color	Varchar(30)	Màu sắc	M

20.Bảng taskpriority: Độ ưu tiên của công việc

VD : immediately, urgent, high , normal , low

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	priorityid	Bigint(20)	PK	M
2	name	Varchar(30)	Tên	M

21.Bảng taskseverity : Độ phức tạp của công việc

VD : critical, major, normal, minor, trivial, enhancement

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	severityid	Bigint(20)	PK	M
2	name	Varchar(30)	Tên	M

22.Bảng group : Nhóm quản lý dự án

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	groupid	Bigint(20)	PK	M
2	leader	Bigint(20)	FK	M

(\*) Mỗi dự án sẽ thuộc về một nhóm, nhóm này được xác định bởi một nhân viên duy nhất và nhân viên này có vai trò Group Leader trong dự án.

23.Bảng roleapplicationpermission : Sự cho phép của các role đối với các ứng dụng

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	roleid	Bigint(20)	PK	M
2	applicationid	Bigint(20)	PK	M
3	permission	Int(11)		M

(\*) permission : 0 – read , 1 : read/write

24.Bảng function : Chức năng của các nhân viên trong dự án

VD : GL, PM, Project Leader, Quality Agent, Developer, Tester...

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	functionid	Bigint(20)	PK	M
2	name	Varchar(30)	Tên	M

25.Bảng employeefunctioninproject : Nhân viên &amp; chức năng của họ trong dự án

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	employeeid	Bigint(20)	PK	M
2	functionid	Bigint(20)	PK	M
3	projectid	Bigint(20)	PK	M

26.Bảng log : Ghi lại toàn bộ quá trình thay đổi trên các công việc của nhân viên

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	taskid	Bigint(20)	PK – Công việc	M
2	time	Datetime	PK – Thời điểm	M
3	employeeid	Bigint(20)	FK – Nhân viên	M
4	reason	Text	Lý do	O
5	field	Varchar(100)	Trường bị thay đổi	M
6	change	Varchar(100)	Nội dung thay đổi	M

27.Bảng taskchild : Sự phụ thuộc giữa các công việc

STT	Tên	Kiểu	Ý nghĩa	Ghi chú
1	taskid	Bigint(20)	PK – Công việc cha	M
2	taskchildid	Bigint(20)	PK – Công việc con	M

### III.Mô hình cơ sở dữ liệu :

## CHƯƠNG V MODULE PHÂN CHIA & PHÂN CÔNG CÔNG VIỆC

### I. Giới thiệu chương trình :

STT	Nội dung
Ứng dụng	Web Application
Ngôn ngữ lập trình	Java
Server	Apache Tomcat 6
IDE	Eclipse Helios
Cơ sở dữ liệu	MySQL Server 5.1, MySQL Workbench 5.2 CE
Công cụ	Maven 2, Power Designer 12.5
Thư viện	Hibernate, Spring MVC , jQuery 1.6, jQuery UI 1.8
Chức năng	+Thêm/Xóa/Sửa dự án, tìm kiếm dự án theo tên, mã số, khách hàng, trạng thái, nhóm +Phân bổ nhân viên vào dự án, xóa nhân viên ra khỏi dự án, tìm kiếm thông tin nhân viên trong dự án theo tên, vai trò, bằng cấp +Tạo công việc, phân công công việc, cập nhật thông tin công việc, tìm kiếm công việc theo nhiều tiêu chuẩn như người tạo, người nhận, trạng thái, độ ưu tiên... +Đăng nhập, phân quyền người dùng +Giao diện được hỗ trợ tiếng Việt và tiếng Anh

Chương trình được chia làm 2 layer :

♦Core layer: Bao gồm các lớp Java persistence, Data Access Object – DAO, Data Transfer Object – DTO và lớp Service. Ngoài ra còn có một số lớp utility khác. Layer này có nhiệm vụ làm việc với cơ sở dữ liệu ứng dụng.

♦Web layer : Layer này được chia làm 2 phần

-Jar : Chứa đựng các Controller , Validator của Spring MVC , các lớp Java bean làm nhiệm vụ phân trang, upload file hay định dạng mã hóa ký tự (UTF-8,ISO...).

-War : Bao gồm toàn bộ các file XML cấu hình của Hibernate và Spring MVC, các trang JSP, JavaScript, một số thư viện như JQuery, JQuery UI, CSS, hình ảnh và các thư viện thể.

Về giao diện, chương trình được hỗ trợ 2 ngôn ngữ : tiếng Việt và tiếng Anh. Chương trình đã thực hiện được việc đa ngôn ngữ trên giao diện. Việc đa ngôn ngữ trong dữ liệu sẽ được thực hiện trong thời gian sắp tới.

Ngoài việc quản lý thông tin các dự án, chương trình sẽ cung cấp thêm một số chức năng khác như quản lý thông tin cá nhân, thông tin người dùng, khách hàng, nhân viên của công ty. Tùy theo vai trò của người dùng mà chương trình sẽ cung cấp các danh mục ứng dụng khác nhau. Hiện tại, chương trình thực đã thực hiện được chức năng quản lý thông tin dự án, các chức năng khác cũng sẽ lần lượt được hiện thực trong thời gian sắp tới.

Trong danh sách người dùng, sẽ có một user đóng vai trò là admin, user này được phép quản lý những công việc chung của ứng dụng như Thêm/Xóa/Sửa dự án, nhân viên, khách hàng, người dùng khác.

Trong ứng dụng quản lý thông tin dự án. Mỗi dự án sẽ có một Group Leader - GL, người này có thể làm leader ở nhiều dự án khác, GL sẽ làm việc trực tiếp với Project Manager - PM. Bên dưới PM là Project Leader – PL. PL là người quản lý mọi công việc trong dự án như tạo công việc, giao việc cho những thành viên khác, thống kê công việc.

Về việc cập nhật thông tin công việc, chỉ có PL và thành viên được giao công việc (hay chủ nhân của công việc) mới có thể cập nhật thông tin cho các công việc đó. Điều đó cũng có nghĩa rằng, các thành viên cũng có thể giao nhiệm vụ của mình cho người khác. Nhưng mọi hành động diễn ra trên mỗi công việc đều được lưu lại một cách chi tiết. Sau đây là một số hình ảnh về chương trình.

## II. Hình ảnh Demo :

### 1. Trang chủ :

Tại trang chủ, user có thể đăng nhập vào hệ thống, thay đổi ngôn ngữ hoặc xem các thông tin chung về công ty như lĩnh vực hoạt động chính, cơ hội nghề nghiệp và các công nghệ mà công ty đang sử dụng.

**Đăng Nhập**

Tên Đăng Nhập

Mật Khẩu

☐ Duy trì trạng thái đăng nhập

[? Quên Mật Khẩu](#)

**BANNER AD**  
MORE INFORMATION

### Chúng tôi là ai?

Được thành lập năm 2003, Công ty VSoft chuyên cung cấp dịch vụ gia công phần mềm và tư vấn giải pháp công nghệ thông tin cho các doanh nghiệp trong và ngoài nước. Đội ngũ kỹ sư trẻ, năng động với kiến thức chuyên sâu về các công nghệ và quy trình phát triển phần mềm, kết hợp với bề dày kinh nghiệm tích lũy được qua việc thực hiện và triển khai các dự án đảm bảo VSoft luôn là một đối tác tin cậy và là sự lựa chọn tốt nhất. Sự tin cậy, chất lượng cao và giá cả cạnh tranh là những lợi thế vượt trội giúp VSoft đã và đang ngày càng khẳng định mình như là một trong những công ty hàng đầu trong lĩnh vực gia công phần mềm tại Việt Nam.

### VSOF - Sự lựa chọn của bạn?

- Thành công trong việc cung cấp những dịch vụ gia công phần mềm cho các khách hàng lớn ở thị trường Mỹ, Anh và Việt Nam
- Đội ngũ quản lý nhiệt huyết, giàu kinh nghiệm, gắn kết và đáng tin cậy
- Nắm vững và áp dụng hiệu quả các quy trình phát triển phần mềm: RUP, SCRUM, Agile Development, PMBOK
- Không ngừng mở rộng và tìm kiếm cơ hội mới
- Tập thể hơn 70 kỹ sư giàu kiến thức và kinh nghiệm
- Tọa lạc ở vị trí thuận lợi
- Áp dụng tiêu chuẩn ISO 9001:2000 trong hệ thống quản lý chất lượng
- Khả năng tổ chức và quản lý cao
- Hiệu suất làm việc cao
- Thành thạo về chuyên môn
- Luôn cập nhật công nghệ mới
- Phương pháp quản lý hiệu quả
- Giá cả cạnh tranh

Lĩnh Vực | Nghề Nghiệp | Công Nghệ

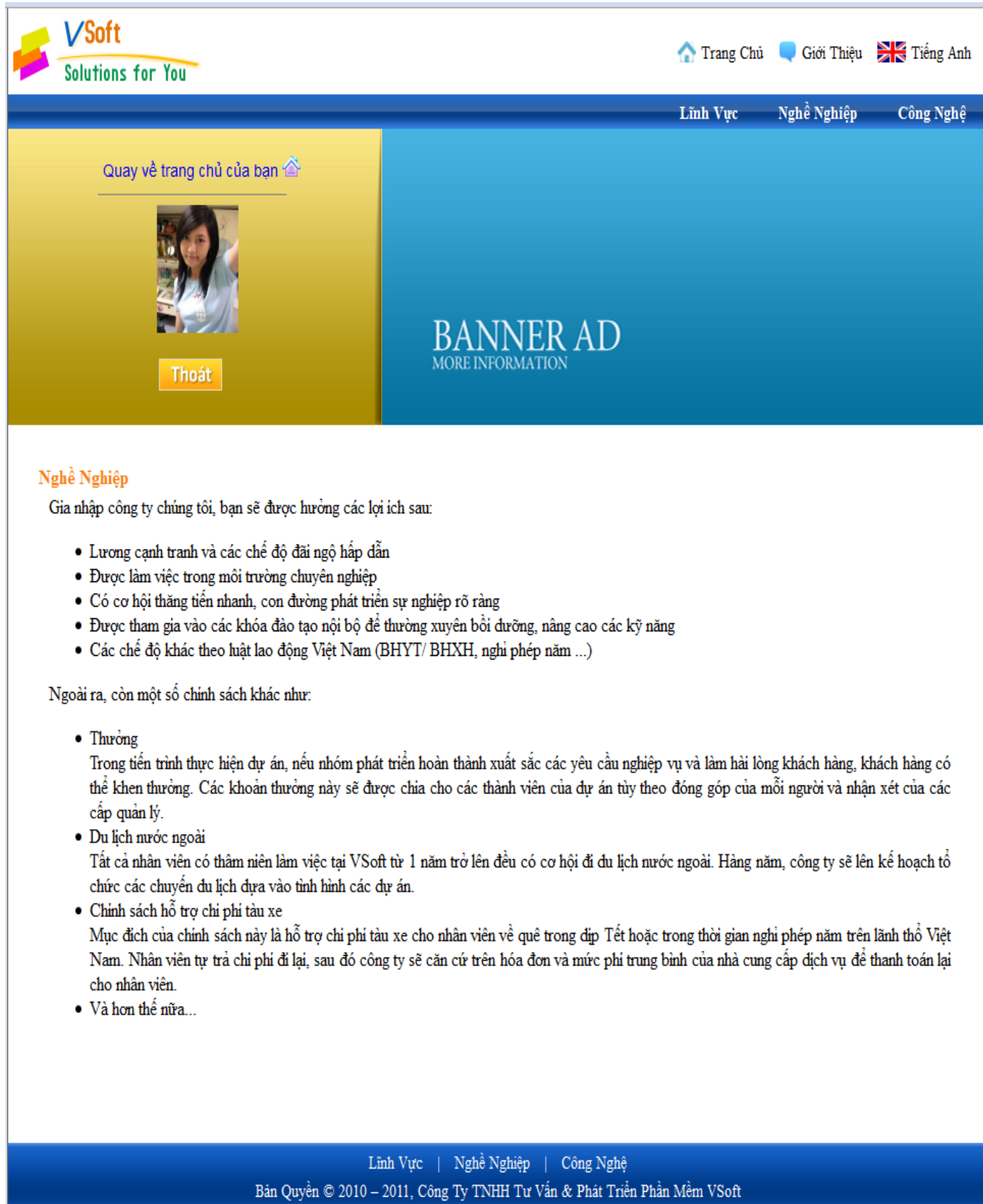
Bản Quyền © 2010 – 2011, Công Ty TNHH Tư Vấn & Phát Triển Phần Mềm VSoft

Hình 5.1 – Trang chủ khi chưa đăng nhập

Khi user click vào “Giới Thiệu”, chương trình sẽ chuyển đến trang chứa một video giới thiệu về công ty. Đây là mới chỉ là ý tưởng, hiện tại tính năng này chưa được thực thi. Hai tính năng “Duy trì trạng thái đăng nhập” và “Quên mật khẩu” cũng sẽ được hiện thực sau.



Trang chủ sau khi user đã đăng nhập :

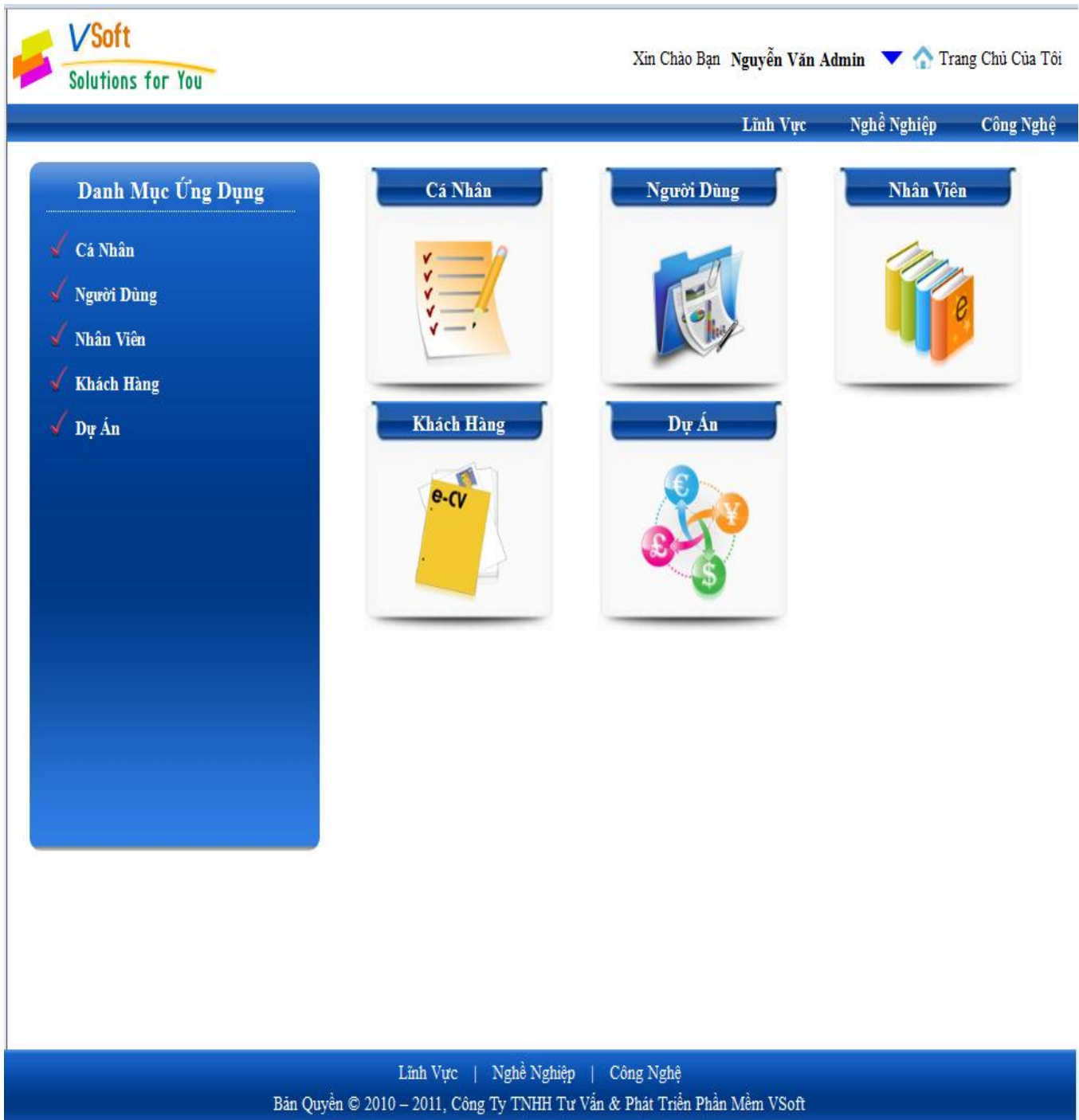


Hình 5.2 – Trang chủ khi đã đăng nhập

Mỗi user sẽ có một hình ảnh đại diện, khi click vào hình ảnh này sẽ đưa user đến trang thông tin cá nhân.

## 2.Trang danh sách các ứng dụng :

Sau khi đăng nhập thành công, chương trình sẽ chuyển đến trang “Trang Chủ Của Tôi” chứa danh sách các ứng dụng.



Hình 5.3 – Danh sách các ứng dụng

Đây là trang ứng dụng của user admin, đối với user khác thì không có ứng dụng “Người Dùng”. Hiện tại ứng dụng “Dự Án” đã được thực hiện, các ứng dụng khác còn đang trong thời gian xây dựng.

Khi user click vào mũi tên màu xanh, một bảng menu sẽ hiện ra cho phép user chọn thay đổi mật khẩu hay thoát khỏi ứng dụng. Khi user chọn thoát khỏi ứng dụng, chương trình sẽ chuyển đến trang chủ ở trạng thái khi chưa đăng nhập.

Khi user click vào dòng chữ tên của user thì chương trình sẽ chuyển đến trang thông tin cá nhân của user đó.

(\*) : Tính năng thay đổi mật khẩu chưa được hiện thực

**3.Trang ứng dụng “Dự Án” :****a) Đối với người dùng là admin:**

Chương trình sẽ hiển thị toàn bộ danh sách các dự án có trạng thái chưa hoàn thành cùng với các nút lệnh Thêm/Xóa/Sửa, tìm kiếm dự án.

**VSoft Solutions for You**

Xin Chào Bạn **Nguyễn Văn Admin** ▾ Trang Chủ Của Tôi

Lĩnh Vực    Nghề Nghiệp    Công Nghệ

**Danh Mục Ứng Dụng**

- ✓ Cá Nhân
- ✓ Người Dùng
- ✓ Nhân Viên
- ✓ Khách Hàng
- ✓ **Dự Án**

**Danh Sách Các Dự Án**

[+ Thêm Dự Án](#) | [Xóa Dự Án](#)

<input type="checkbox"/>		<a href="#">Sửa</a> <b>ProGate</b>	Mã Số : 345 Nhóm : vydx Ngày Bắt Đầu : 2011-08-31	Khách Hàng : HP Trạng Thái : Invalidate
<input type="checkbox"/>		<a href="#">Sửa</a> <b>ProSim</b>	Mã Số : 34 Nhóm : vydx Ngày Bắt Đầu : 2011-07-23	Khách Hàng : IBM Trạng Thái : Invalidate

Hiện Thị 1 - 2 / 2 Kết Quả    Số Item Mỗi Trang     Trang     [|| Đầu](#)   [◀ Trước](#)   [Sau ▶](#)   [Cuối ||](#)

Lĩnh Vực | Nghề Nghiệp | Công Nghệ

Bản Quyền © 2010 – 2011, Công Ty TNHH Tư Vấn & Phát Triển Phần Mềm VSoft

Hình 5.4 – Danh sách dự án của admin

(\*) : Chức năng phân trang chỉ mới được hiện thực một phần.

Trang “Thêm Dự Án” :

**VSoft**  
Solutions for You

Xin Chào Bạn **Nguyễn Văn Admin** ▾ Trang Chủ Của Tôi

Lĩnh Vực    Nghề Nghiệp    Công Nghệ

**Danh Mục Ứng Dụng**

- ✓ Cá Nhân
- ✓ Người Dùng
- ✓ Nhân Viên
- ✓ Khách Hàng
- ✓ Dự Án

**Thêm Dự Án**

(\*) Thông Tin Bắt Buộc Nhập

Tên Dự Án (\*)

Mã Số (\*)

Khách Hàng (\*)

Nhóm (\*)

Trạng Thái (\*)

Ngày Bắt Đầu (\*)

Ngày Kết Thúc


Lĩnh Vực | Nghề Nghiệp | Công Nghệ


Bản Quyền © 2010 – 2011, Công Ty TNHH Tư Vấn & Phát Triển Phần Mềm VSoft

Hình 5.5 – Thêm dự án

User có thể thêm nhiều dự án, nhưng các dự án phải có mã số khác nhau. Các trường có đánh dấu (\*) Là những trường bắt buộc phải nhập. Trường ngày kết thúc dự án không bắt buộc nhưng nếu chọn thì nó phải có giá trị sau này bắt đầu dự án.

Trang “Sửa Dự Án” :



Xin Chào Bạn **Nguyễn Văn Admin** ▾  Trang Chủ Của Tôi

Lĩnh Vực    Nghề Nghiệp    Công Nghệ

**Danh Mục Ứng Dụng**

✓ Cá Nhân

✓ Người Dùng

✓ Nhân Viên

✓ Khách Hàng

✓ Dự Án

**Cập nhật Dự Án**

(\*) Thông Tin Bắt Buộc Nhập

Tên Dự Án (\*)

ProGate

Mã Số (\*)

345

Khách Hàng (\*)

HP ▾


Nhóm (\*)

vydx ▾


Trạng Thái (\*)

Invalidate ▾

Ngày Bắt Đầu (\*)

2011-08-31 

Ngày Kết Thúc

2011-10-31 

Lưu

Hủy

Lĩnh Vực | Nghề Nghiệp | Công Nghệ

Bản Quyền © 2010 – 2011, Công Ty TNHH Tư Vấn & Phát Triển Phần Mềm VSoft

Hình 5.6 – Sửa dự án

**b) Đối với người dùng khác :**

Chương trình sẽ hiển thị những danh sách dự án có trạng thái chưa hoàn thành mà người dùng đó đang tham gia vào.

**VSoft Solutions for You**



Xin Chào Bạn **Đông Xuân Vỹ** ▾ Trang Chủ Của Tôi

Lĩnh Vực    Nghề Nghiệp    Công Nghệ

**Danh Mục Ứng Dụng**

- ✓ Cá Nhân
- ✓ Nhân Viên
- ✓ Khách Hàng
- ✓ **Dự Án**

**Danh Sách Các Dự Án**

	<b>ProGate</b> Mã Số : 345 Nhóm : vydx Ngày Bắt Đầu : 2011-08-31 Khách Hàng : HP Trạng Thái : Invalidate
	<b>ProSim</b> Mã Số : 34 Nhóm : vydx Ngày Bắt Đầu : 2011-07-23 Khách Hàng : IBM Trạng Thái : Invalidate

Lĩnh Vực | Nghề Nghiệp | Công Nghệ

Bản Quyền © 2010 – 2011, Công Ty TNHH Tư Vấn & Phát Triển Phần Mềm VSoft

Hình 5.7 – Danh sách dự án của người dùng bình thường

**4)Trang “Trang Chủ Dự Án” :**

Khi user chọn dự án nào trong danh sách dự án ở trên, chương trình sẽ chuyển đến trang chủ của dự án đó. Tại trang chủ của mỗi dự án, hiện tại chương trình cung cấp hai chức năng :

+Thông tin về các thành viên trong dự án

+Cho phép các thành viên trong dự án đi vào khu vực làm việc của mình (workspace)

**a) Thông tin thành viên :**

Chức năng này cung cấp cho cả admin và các thành viên khác trong dự án.

Danh sách thành viên

**VSoft Solutions for You**

Xin Chào Bạn **Nguyễn Văn Admin** ▾ Trang Chủ Của Tôi

Lĩnh Vực | Nghề Nghiệp | Công Nghệ

Danh Sách Thành Viên | Thêm Thành Viên [Quay Về Trang Chủ Dự Án]

**Danh Sách Thành Viên**

Thứ Hạng:  Văn Bảng:

STT	<input type="checkbox"/>	Họ & Tên	Visa	Thứ Hạng	Văn Bảng	Chức Năng
1	<input type="checkbox"/>	Lê Kiều Anh	anhlk	Engineer	Kỹ sư	Developer
2	<input type="checkbox"/>	Bùi Tuấn Em	embt	Engineer	Kỹ sư	Tester
3	<input type="checkbox"/>	Phan Tuấn Hải	haipt	Engineer	Kỹ sư	Projec Manager
4	<input type="checkbox"/>	Đông Xuân Vỹ	vydx	Engineer	Kỹ sư	Project Leader
5	<input type="checkbox"/>	Nguyễn Quang Đạo	daonq	Engineer	Kỹ sư	Developer

Hiện Thị 1 - 5 / 5 Kết Quả


Số Item Mỗi Trang  Trang



Lĩnh Vực | Nghề Nghiệp | Công Nghệ

Bản Quyền © 2010 – 2011, Công Ty TNHH Tư Vấn & Phát Triển Phần Mềm VSoft

Hình 5.8 – Danh sách thành viên trong dự án

Admin có quyền xóa thành viên ra khỏi dự án.



Xin Chào Bạn **Nguyễn Văn Admin**

 Trang Chủ Của Tôi

Lĩnh Vực
Nghề Nghiệp
Công Nghệ

Danh Sách Thành Viên
Thêm Thành Viên

[\[Quay Về Trang Chủ Dự Án\]](#)

### Danh Sách Thành Viên

Thứ Hạng

Tất Cả

Văn Bằng

Tất Cả

Tìm kiếm

STT	<input type="checkbox"/>	Họ & Tên	Văn Bằng	Chức Năng
1	<input checked="" type="checkbox"/>	Lê Kiều Anh		Developer
2	<input checked="" type="checkbox"/>	Bùi Tuấn Em		Tester
3	<input type="checkbox"/>	Phan Tuấn Hải		Projec Manager
4	<input type="checkbox"/>	Đông Xuân Vỹ		Project Leader
5	<input type="checkbox"/>	Nguyễn Quang Đạo	daonq Engineer K.y sư	Developer

Thông báo

Bạn có đồng ý xóa (những) thành viên này khỏi dự án không?

Đồng ý

Hủy

Xóa

Hủy

Hiện Thị 1 - 5 / 5 Kết Quả

Số Item Mỗi Trang 10
Trang 1

Đầu
Trước
Sau
Cuối


Lĩnh Vực | Nghề Nghiệp | Công Nghệ



Bản Quyền © 2010 – 2011, Công Ty TNHH Tư Vấn & Phát Triển Phần Mềm VSoft

Hình 5.9 – Xóa thành viên ra khỏi dự án



Admin có quyền thêm thành viên vào dự án .



Xin Chào Bạn **Nguyễn Văn Admin**

 Trang Chủ Của Tôi

Lĩnh Vực
Nghề Nghiệp
Công Nghệ

Danh Sách Thành Viên
Thêm Thành Viên

[\[Quay Về Trang Chủ Dự Án\]](#)

**Thêm Thành Viên**

Thứ Hạng

Tất Cả

Văn Bằng

Tất Cả

Tìm kiếm

STT	<input type="checkbox"/>	Họ & Tên	Visa	Thứ Hạng	Văn Bằng	Chức Năng
1	<input checked="" type="checkbox"/>	Đặng Quang Huy				Developer
2	<input checked="" type="checkbox"/>	Triệu Sùng Lý				Developer
3	<input type="checkbox"/>	Đặng Phú Quý				Vui Lòng Chọn

Thông báo

Bạn có đồng ý thêm (những) thành viên này vào dự án không?

Đồng ý

Hủy

Hiện Thị 1 - 3 / 3 Kết Quả

Trang 1

Đầu
Trước
Sau
Cuối

Lĩnh Vực
|
Nghề Nghiệp
|
Công Nghệ


Bản Quyền © 2010 – 2011, Công Ty TNHH Tư Vấn & Phát Triển Phần Mềm VSoft



Hình 5.10 – Thêm thành viên vào dự án

**b)Workspace :**

Chức năng này chỉ cung cấp cho các thành viên trong dự án. Trang này được thiết kế một menu bao gồm : Danh Sách Nhiệm Vụ, Nhiệm Vụ Của Tôi, Thêm Nhiệm Vụ.

•Danh Sách Nhiệm Vụ : Liệt kê ra 10 nhiệm vụ có thời gian tạo ra gần với thời điểm hiện tại nhất, thuộc các thể loại sau : Nhiệm Vụ Mới, Nhiệm Vụ Đã Giao, Nhiệm Vụ Chưa Hoàn Thành, Nhiệm Vụ Hoàn Thành. Tùy thuộc vào vai trò của user trong dự án mà các thể loại công việc được hiển thị sẽ khác nhau.







Xin Chào Bạn **Đồng Xuân Vỹ**   Trang Chủ Của Tôi

Lĩnh Vực | Nghề Nghiệp | Công Nghệ

[Danh Sách Nhiệm Vụ](#) | [Nhiệm Vụ Của Tôi](#) | [Thêm Nhiệm Vụ](#)

[Quay Về Trang Chủ Dự Án]

<b>Nhiệm Vụ Mới (1 - 1/1)</b> <div> <a href="#">345004</a>  [Org Profile]: Quản Lý Người Dùng / Root Org / Phân Quyền: Owner không được phân quyền Owner cho người dùng khác Bug - 2011-09-03 09:27         </div>	<b>Nhiệm Vụ Đã Giao (1 - 1/1)</b> <div> <a href="#">345005</a>  [Organization Profile] - Sửa Tin Tức Sự Kiện Nổi Bật: Hệ thống không hiển thị ảnh trước đó Bug - 2011-09-03 09:28         </div>
<b>Nhiệm Vụ Hoàn Thành (1 - 2/2)</b> <div> <a href="#">345002</a>  [Organization Profile] - Hiện thực nút "Đặt mua" cho từng sản phẩm Bug - 2011-09-03 09:26         </div> <div> <a href="#">345000</a>  [Organization Management] - Không xóa được tổ chức ra khỏi database Bug - 2011-09-03 09:23         </div>	

new

assigned

confirmed

feedback

pending

to be reviewed

resolved


closed


[Danh Sách Nhiệm Vụ](#) | [Nhiệm Vụ Của Tôi](#) | [Thêm Nhiệm Vụ](#)

Lĩnh Vực | Nghề Nghiệp | Công Nghệ

Bản Quyền © 2010 – 2011, Công Ty TNHH Tư Vấn & Phát Triển Phần Mềm VSoft

Hình 5.11 – Danh sách công việc đối với GL, PM và PL









Xin Chào Bạn **Lê Kiều Anh** ▼  Trang Chủ Của Tôi

[Lĩnh Vực](#)   [Nghề Nghiệp](#)   [Công Nghệ](#)

[Danh Sách Nhiệm Vụ](#) | [Nhiệm Vụ Của Tôi](#)

[\[Quay Về Trang Chủ Dự Án\]](#)

Nhiệm Vụ Chưa Xong (1 - 3/3)		Nhiệm Vụ Mới (1 - 1/1)	
<a href="#">345005</a> 	[Organization Profile] - Sửa Tin Tức Sự Kiện Nổi Bật: Hệ thống không hiển thị ảnh trước đó Bug - 2011-09-03 09:28	<a href="#">345004</a> 	[Org Profile]: Quản Lý Người Dùng / Root Org / Phân Quyền: Owner không được phân quyền Owner cho người dùng khác Bug - 2011-09-03 09:27
<a href="#">345003</a> 	[Organization Profile] - Hỗ Trợ: Hệ thống không giới hạn dung lượng file PDF Bug - 2011-09-03 09:26		
<a href="#">345001</a> 	[Organization Profile] - Tích hợp CDS và LMIS vào ProGate Bug - 2011-09-03 09:25		

Nhiệm Vụ Hoàn Thành (1 - 2/2)	
<a href="#">345002</a> 	[Organization Profile] - Hiện thực nút "Đặt mua" cho từng sản phẩm Bug - 2011-09-03 09:26
<a href="#">345000</a> 	[Organization Management] - Không xóa được tổ chức ra khỏi database Bug - 2011-09-03 09:23

[new](#)

[assigned](#)

[confirmed](#)

[feedback](#)

[pending](#)

[to be reviewed](#)

[resolved](#)

[closed](#)

[Danh Sách Nhiệm Vụ](#) | [Nhiệm Vụ Của Tôi](#)

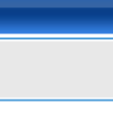
[Lĩnh Vực](#) | [Nghề Nghiệp](#) | [Công Nghệ](#)

Bản Quyền © 2010 – 2011, Công Ty TNHH Tư Vấn & Phát Triển Phần Mềm VSoft

Hình 5.12 – Danh sách công việc đối với user khác

Ngay sau danh sách công việc là menu màu sắc của các trạng thái của bất kỳ một công việc nào trong dự án.

Đường link tại mã số công việc sẽ dẫn đến trang chi tiết về công việc đó.



VSoft

Solutions for You

Xin Chào Bạn

Lê Kiều Anh

▼

Trang Chủ Của Tôi

Lĩnh Vực

Nghề Nghiệp

Công Nghệ

Danh Sách Nhiệm Vụ | Nhiệm Vụ Của Tôi

[Quay Về Trang Chủ Dự Án]

Thông Tin Chi Tiết

Mã Số	Phân Loại	Độ Phức Tạp	Thời Gian Tạo	Cập Nhật Cuối
345001	Bug	normal	2011-09-03 09:25	
Người Tạo	vydx			
Người Nhận	anhlk			
Độ Ưu Tiên	normal			
Trạng Thái	to be reviewed			
Tóm Tắt	[Organization Profile] - Tích hợp CDS và LMIS vào ProGate			
Miêu Tả	- Tích hợp CDS và LMIS vào ProGate			
Thông Tin Thêm				
% Hoàn Thành	0 %			
Số Giờ Thực Tế				
Số Giờ Ước Lượng				
Ngày Bắt Đầu	2011-09-03			
Pha				
Ghi Chú				
Cập Nhật	Quay Lại			

☐ Lịch Sử

Thời Gian	Visa	Nội Dung	Thay Đổi

Danh Sách Nhiệm Vụ | Nhiệm Vụ Của Tôi

Lĩnh Vực | Nghề Nghiệp | Công Nghệ


Bản Quyền © 2010 – 2011, Công Ty TNHH Tư Vấn & Phát Triển Phần Mềm VSoft

### Hình 5.13 –Chi tiết công việc

Bảng “Lịch Sử” sẽ hiển thị tất cả các thay đổi mà các thành viên trong dự án đã tạo ra trên công việc.

Nút lệnh “Cập Nhật” cho phép user có thể chuyển đến trang cập nhật thông tin công việc. Ngoài ra user cũng có thể click vào biểu tượng có cây viết ở danh sách nhiệm vụ phía trên khi muốn cập nhật thông tin về công việc. Và chú ý rằng, chỉ có PL và người được giao công việc này mới có thể cập nhật nó mà thôi.

Trang cập nhật công việc.



Xin Chào Bạn **Đông Xuân Vỹ**

▼

Trang Chủ Của Tôi

Lĩnh Vực

Nghề Nghiệp

Công Nghệ

[Danh Sách Nhiệm Vụ](#) | [Quản Lý Nhiệm Vụ](#) | [Thêm Nhiệm Vụ](#)

[\[Quay Về Trang Chủ Dự Án\]](#)

**Cập Nhật Thông Tin Nhiệm Vụ**

**Cập nhật nhiệm vụ thành công**

Mã Số	Phân Loại (*)	Độ Phức Tạp (*)	Thời Gian Tạo	Cập Nhật Cuối
345000	Bug	normal	2011-09-03 09:23	
Người Tạo	vydx			
Người Nhận	anhlk			
Độ Ưu Tiên (*)	normal			
Trạng Thái (*)	assigned			
Tóm Tắt (*)	[Organization Management] - Không xóa được tổ chức ra khỏi database			
Miêu Tả (*)	- <u>Không</u> <u>xóa</u> được tổ chức <u>ra</u> <u>khỏi</u> database			
Thông Tin Thêm				
% Hoàn Thành				
Số Giờ Thực Tế				
Số Giờ Ước Lượng				
Ngày Bắt Đầu (*)	2011-09-03			
Pha				
Ghi Chú				

Lưu

Hủy

[Danh Sách Nhiệm Vụ](#) | [Quản Lý Nhiệm Vụ](#) | [Thêm Nhiệm Vụ](#)

Lĩnh Vực

Nghề Nghiệp

Công Nghệ

Bản Quyền © 2010 – 2011, Công Ty TNHH Tư Vấn & Phát Triển Phần Mềm VSoft

Hình 5.14 –Cập nhật công việc

●Nhiệm Vụ Của Tôi :

Cho phép user tìm kiếm công việc của mình hoặc của các thành viên khác theo nhiều tiêu chuẩn khác nhau.



Xin Chào Bạn **Lê Kiều Anh** ▾  Trang Chủ Của Tôi

Lĩnh Vực    Nghề Nghiệp    Công Nghệ

[Danh Sách Nhiệm Vụ](#) | [Nhiệm Vụ Của Tôi](#)

[Quay Về Trang Chủ Dự Án]

Người Tạo	Người Nhận	Phân Loại	Độ Phức Tạp
[Bất Kỳ] ▾	anhlk ▾	[Bất Kỳ] ▾	[Bất Kỳ] ▾
Độ Ưu Tiên	Hiện Trạng Thái	Ẩn Trạng Thái	Liên Quan
[Bất Kỳ] ▾	[Bất Kỳ] ▾	[Bất Kỳ] ▾	[Bất Kỳ] ▾
Pha	Ngày Bắt Đầu	Hiện Thị	
<input type="text"/>	<input type="text"/> 	[Bất Kỳ] ▾	
Lọc	Reset		

**Danh Sách Nhiệm Vụ (1 - 3/3)**

	Mã Số	Phân Loại	Độ Phức Tạp	Độ Ưu Tiên	Trạng Thái	Tóm Tắt
	<a href="#">345005</a>	Bug	normal	normal	assigned (anhlk)	[Organization Profile] - Sửa Tin Tức Sự Kiện Nổi Bật: Hệ thống không hiển thị ảnh trước đó
	<a href="#">345003</a>	Bug	normal	normal	to be reviewed (anhlk)	[Organization Profile] - Hỗ Trợ Hệ thống không giới hạn dung lượng file PDF
	<a href="#">345001</a>	Bug	normal	normal	to be reviewed (anhlk)	[Organization Profile] - Tích hợp CDS và LMIS vào ProGate

[Danh Sách Nhiệm Vụ](#) | [Nhiệm Vụ Của Tôi](#)

Lĩnh Vực | Nghề Nghiệp | Công Nghệ

Bản Quyền © 2010 – 2011, Công Ty TNHH Tư Vấn & Phát Triển Phần Mềm VSoft

Hình 5.15 –Nhiệm vụ của tôi

- Thêm Công Việc :  
Chức năng này chỉ dành cho PL.

VSoft

Solutions for You

Xin Chào Bạn **Đông Xuân Vỹ**

Trang Chủ

Của Tôi

Lĩnh Vực

Nghề Nghiệp

Công Nghệ

Thêm Nhiệm Vụ

✓ Thêm nhiệm vụ thành công

(\*) Thông Tin Bắt Buộc Nhập

Phân Loại (\*)

Bug

Độ Phức Tạp (\*)

normal

Độ Ưu Tiên (\*)

normal

Tóm Tắt (\*)

[Organization Management] - Không xóa được tổ chức ra khỏi database

Miêu Tả (\*)

- Không xóa được tổ chức ra khỏi database

Thông Tin Thêm

Số Giờ Ước Lượng

Ngày Bắt Đầu (\*)

2011-09-03

Pha

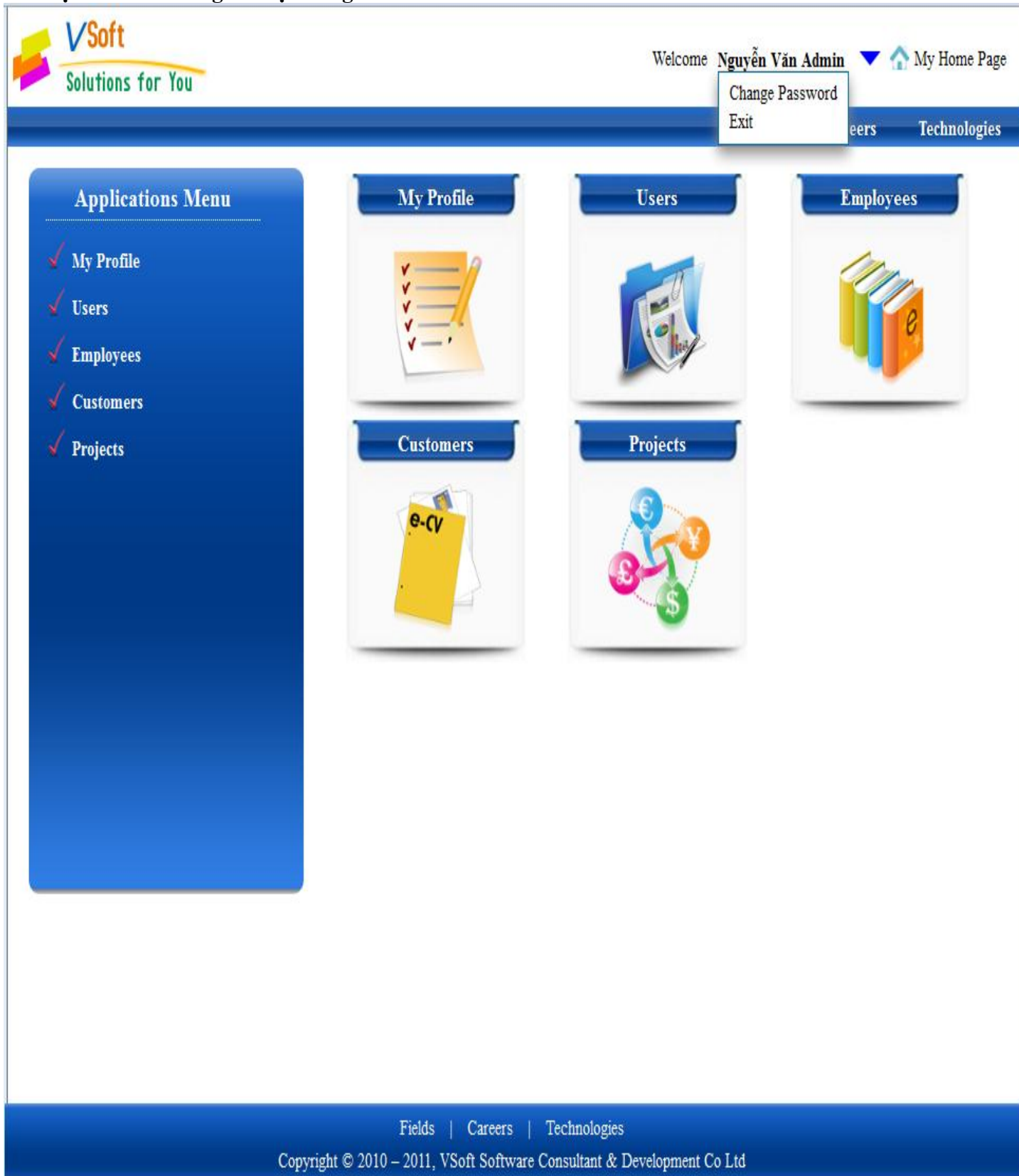
Lưu

Hủy

Lĩnh Vực | Nghề Nghiệp | Công Nghệ


Bản Quyền © 2010 – 2011, Công Ty TNHH Tư Vấn & Phát Triển Phần Mềm VSoft

### Hình 5.16 – Thêm công việc

**5.Một vài hình ảnh giao diện tiếng Anh:**

Hình 5.17 – “Trang Chủ Của Tôi” giao diện tiếng Anh





Welcome **Nguyễn Văn Admin**
[My Home Page](#)



[Fields](#)
[Careers](#)
[Technologies](#)

Applications Menu

- ✓ My Profile
- ✓ Users
- ✓ Employees
- ✓ Customers
- ✓ **Projects**

Projects

[+ Add Project](#) | [Delete Projects](#)

<input type="checkbox"/> 	<a href="#">Edit</a> <b>ProGate</b> Number : 345 Group : vydx Start Date : 2011-08-31 Customer : HP Status : Invalidate
<input type="checkbox"/> 	<a href="#">Edit</a> <b>ProSim</b> Number : 34 Group : vydx Start Date : 2011-07-23 Customer : IBM Status : Invalidate

Show 1 - 2 / 2 Results
 Items/Page 
 Page 
[First](#)
[Previous](#)
[New](#)
[Last](#)

[Fields](#) | [Careers](#) | [Technologies](#)  
 Copyright © 2010 – 2011, VSoft Software Consultant & Development Co Ltd

Hình 5.18 – Danh sách dự án giao diện tiếng Anh

## KẾT LUẬN

Có thể nói chương trình quản lý thông tin các dự án phần mềm là một chương trình có khả năng mở rộng là rất lớn. Nhưng do thời gian có giới hạn nên em chỉ thực hiện được một phần nhỏ chức năng của chương trình. Trong thời gian tới đây, em sẽ cố gắng hiện thực được những chức năng còn lại để chương trình ngày càng được hoàn thiện hơn.

Sau hơn 4 tháng làm việc trong Java, đặc biệt là với Hibernate và Spring MVC. Em đã hiểu rõ về 2 framework này hơn rất nhiều. Thực sự khi mới tiếp cận với 2 framework này, không ít người sẽ cảm thấy khó khăn bởi lẽ trong chúng cái gì cũng mới lạ. Nhưng khi đã hiểu được chúng rồi mới thấy chúng thật hay. Những dòng code thật ngắn gọn, những tính năng mà chúng hỗ trợ...

Trong phần sau của đề tài, em sẽ tìm hiểu và áp dụng các tính năng nâng cao trong 2 framework này như Hibernate Caching hay Spring MVC Annotations nhằm cải thiện tốc độ của chương trình, điều mà bấy lâu nay vẫn là hạn chế của Hibernate.

## DANH MỤC TÀI LIỆU THAM KHẢO

### Tiếng Anh:

1. Hibernate Annotations.
2. Manning - Java Persistence with Hibernate. ©2007 by Manning Publications Co. All rights reserved.
3. Manning.Spring.in.Action.2nd.Edition.Aug.2007. ©2008 by Manning Publications Co. All rights reserved.
4. Expert Spring MVC and Web Flow. Copyright © 2006 by Seth Ladd, Darren Davison, Steven Devijver, and Colin Yates

### Danh mục các Website tham khảo:

1. <http://maven.apache.org/>
2. <http://www.hibernate.org/>
3. <http://www.roseindia.net>
4. <http://www.vaannila.com>
5. <http://docs.jboss.org>
6. <http://www.mkyong.com>